DOCUMENT RESUME

ED 042 368   24   EM 008 474

AUTHOR	Judd, Wilson A.
TITLE	The Development of an On-Line Laboratory for CAI and Behavioral Research (1964-1968). Technical Report.
INSTITUTION	Pittsburgh Univ., Pa. Learning Research and Development Center.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Bureau of Research.
BUREAU NO	BR-5-0253
PUB DATE	69
NOTE	149p.
EDRS PRICE	EDRS Price MF-$0.75 HC-$7.55
DESCRIPTORS	*Behavioral Science Research, Computer Assisted Instruction, *Computer Based Laboratories, *Digital Computers, Display Panels, Display Systems, Documentation, Experimental Programs, Laboratory Equipment, Personnel, Programing Languages, Student Employment, *Time Sharing
IDENTIFIERS	ETSS, *Experimental Time Sharing System, Learning Research and Development Center, LRDC

ABSTRACT

Psychologists and educational researchers who may be interested in installing a computer system in a behavioral science laboratory may learn from the experience gained in the development and operation of the Computer Facility at the Learning and Research Development Center of the University of Pittsburgh. Built around a PDP-7 computer, using a system designed much like an early process control system, the Facility in 1969 developed a new time-sharing system, the LRDC Experimental Time-Sharing System (ETSS). The elements of its hardware are explained: main frame, information transmission device control, automatic priority interrup system, multiplexor, clocks, relays and buffers, PDP-7/PDP-9 interface, input-output devices, subject terminals, and remote terminals. The various aspects of the software are also explained: jobs and job scheduling, master tables, memory management, operator control, input/output device routines, stimulus device control routines, response device control routines, utility programs, and higher-level languages. The physical plant is described and personnel duties listed. A description of 12 experimental control programs is provided. The importance of documentation is emphasized and appendices illustrate documentation undertaken. Practical advice and references are given. (MF)

UNIVERSITY OF PITTSBURGH - LEARNING R & D CENTER

TECHNICAL REPORT

THE DEVELOPMENT OF AN ON-LINE LABORATORY FOR CAI AND BEHAVIORAL RESEARCH (1964-1968)

WILSON A. JUDD

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ED0 42368

EM008 474

ED0 42368

THE DEVELOPMENT OF AN ON-LINE LABORATORY FOR CAI AND BEHAVIORAL RESEARCH (1964-1968)

Wilson A. Judd*

Learning Research and Development Center

University of Pittsburgh

Winter 1969

---

*Now Assistant Professor, Department of Educational Psychology, University of Texas, Austin.

Published by the Learning Research and Development Center supported in part as a research and development center by funds from the United States Office of Education, Department of Health, Education, and Welfare. The opinions expressed in this publication do not necessarily reflect the position or policy of the Office of Education and no official endorsement by the Office of Education should be inferred.

FOREWORD

The Computer Support Facility of the Learning Research and Development Center has three functions: (1) to service laboratory experimentation on processes of learning related to instruction, (2) to support developmental work on the use of computers in education, including instruction, testing, and instructional management, and (3) to conduct design and development work on the requisite software and hardware for carrying out functions (1) and (2).

Preliminary planning of the facility took place during the year 1964-65. William W. Ramage, on part-time loan from the Westinghouse Research Laboratories, provided the expertise required for initial system planning. He, together with other members of the Center staff, identified objectives, drew up hardware specifications, contacted hardware suppliers, and obtained bids for computer components. Also during this year a document was prepared on the requirements for student stations, i.e., student input-output terminals. The report emphasized the considerations involved in designing the "interface between the student and the subject matter" (Glaser, Lipson & Ramage, 1964; Glaser & Ramage, 1967). In the summer of 1965, delivery was made of components for a system built around a Digital Equipment Corporation PDP-7 computer.

The period 1965-67 was a time of system implementation and pilot use, under the guidance of William W. Ramage, an electronics engineer, and Ronald G. Ragsdale, an educational psychologist. During this period, effort included a joint project with the Westinghouse Research Laboratories on the development of interface devices for subject-matter display and student response. It was also a time of software development as described in the following report. The system as then envisioned was described in a 1966 publication.* From 1967-69, the system was in operation and under further development, under the direction of J. G. Castle and later, Wilson A. Judd (William Ramage had returned to full-time study in mathematics). Wilson Judd is especially qualified to write this report. He initally worked with the system in running experiments in verbal learning; later, as director of the facility he was concerned broadly with the development of an on-line computer laboratory for behavioral research.

As well as a history of past work, this report serves as background for new activities taking place in LRDC's Computer Support Facility, under the direction of Robert Fitzhugh. Since the Fall of 1968, the system has undergone a series of major modifications intended to simplify programming and to facilitate its operation in a time-shared mode. Input/output routines were rewritten and unnecessary options eliminated, several internal changes were made which simplified the writing of reentrant programs, new programming conventions were adopted including the use of push-down stacks for subroutine return storage, and a powerful set of operator control routines were added, which provided the computer operator with greater control over the system while it operated in a time-shared mode. These modifications have significantly improved system performance and reliability. In addition, analyses of limitations inherent in the original design have led to work that is of current priority in 1969-70.

To fully understand the current system and its strengths and weaknesses, it is necessary to understand its origins and the intentions of the original designers. First, it must be remembered that the system was designed in 1964 when time-sharing systems were just beginning to be designed elsewhere. Second, and more important, the designers of the system had acquired most of their experience in the area of process control systems. Since the computer requirements of an on-line laboratory are not unlike the requirements of many process control applications, this experience seemed quite relevant. As a result, the system was designed both internally and externally much like an early process control system.

---

*Ronald Ragsdale. The LRDC's Computer-Assisted Laboratory, DECUS Proceedings, February, 1966, 5, 65-68.

Responsiveness is a major consideration in the design of any process control system. The need to be highly responsive generally precludes swapping since the delays that inevitably occur in a swapping system simply cannot be tolerated. Because of this concern, the LRDC system avoided swapping and was designed to execute reentrant programs. In addition, the system was designed to be "event driven" and depended upon external events in the form of hardware interrupts to drive the scheduler. It was felt that this particular design philosophy would ensure that the system would be highly responsive to external demands for service. When the LRDC system is operating, all jobs in the system are initiated and controlled from a single operator's console, rather than from the user terminals as is customary in general-purpose time-sharing systems. This is again a characteristic of process control systems since such systems interact with machines, not individuals.

It might seem that this system would be ideal for an on-line laboratory environment; however, in spite of its many attractive features, the system cannot be considered to be wholly successful in terms of the requirements we aspire to. In process control applications, a package of programs is developed and becomes semi-permanent, and care is taken to ensure that no one program abuses the freedoms of the system. Once a full complement of programs has been developed for a particular application, new programs are added infrequently, and the system tends to stabilize. In an active, on-line laboratory, however, programs are in all stages of development, and new programs are added regularly to the job mix. The researcher's primary interest is in the rapid development and checkout of a new program, followed by its smooth operation along with ample computer time to conduct the particular piece of research. It is in these areas, however, that the current LRDC system falls short. In particular, since the primary design goal of the system was on extensive capabilities rather than on ease of programming, program development time is typically lengthy, even with skilled programmers. All programs must be written in assembly language to meet the reentrant code requirements of the system, and must interface with a complex input/output structure. Only one program can be debugged at a time, and because the entire operation of the system is centered about a single operator's console, debugging cannot occur while other programs are operating. In addition, programs must be subjected to an unusually thorough and lengthy checkout procedure before they reach production since the system is unprotected, and an error in any one program can induce a systems crash.

In late 1969 a decision was made to begin the development of a new time-sharing system based on an entirely different design philosophy. This system has come to be known as the LRDC Experimental Time-Sharing System (ETSS) and is currently undergoing implementation. ETSS is a general-purpose, multi-language, time-sharing system designed to operate on a small- to medium-scale computer equipped with a fast swapping disk. ETSS has been designed to support flexibly a wide variety of real-time applications ranging from highly interactive terminal-oriented tasks to on-line laboratory experimentation where close control of nonstandard devices is required. Ease of programming and program debugging has been stressed. New programs may be debugged while other production programs are operating which should serve to relieve the serious debugging bottleneck that has developed with increased use.

ETSS consists of three main logical units, the MONITOR, the EXECUTIVE, and a number of SUB-SYSTEMS. The EXECUTIVE is the heart of the time-sharing system and performs two major functions. First, it is responsible for the allocation and control of all systems resources including input/output channels, memory, computation time, and space on auxiliary storage devices. Second, the EXECUTIVE provides a set of services to programs operating under its control. By masking hardware idiosyncrasies, the EXECUTIVE enables the user to interface with a "virtual" software machine which embodies abstractions such as "files and records" and which is far more user-oriented than the underlying hardware. Programs running under the control of the EXECUTIVE are referred to as "tasks," which are discrete units of work to be performed by the system. A task might be a program written by a user at a terminal or a system program that is invoked by the EXECUTIVE itself to provide a required service or function.

The MONITOR is the second main logical unit in ETSS and serves as a software interface between the user at a terminal and the EXECUTIVE. Through the MONITOR COMMAND LANGUAGE (MCL), the user gains access to the system (LOGIN), acquires peripherals (ASSIGN), manipulates permanent files and file directories (CATALOG, RENAME, DELETE, DIRECTORY, FILES, LIBRARY), communicates with the operator (TALK), requests system news and time (NEWS, TIME), calls ETSS SUB-SYSTEMS (EDITOR, ASSEMBLER, FORTRAN, LOADER, DMS, DEBUG, FOCAL, BASIC T64), and returns facilities to the system when the job is finished (DEASSIGN, LOGOUT). Both the MONITOR and the EXECUTIVE reside in a protected portion of memory termed SYSPACE.

SUB-SYSTEMS, the third component of ETSS, as well as user programs, reside in the remaining portion of memory called USPACE.

USPACE is partitioned by the EXECUTIVE among users and is swapped to and from a high-speed disk or drum when required. As part of the overall ETSS design philosophy, the core-resident MONITOR provides only a limited set of well-defined services, and most processing occurs in a SUB-SYSTEM located in USPACE. Although overhead is increased since multiple copies of the same SUB-SYSTEM must be swapped, the MONITOR and EXECUTIVE are kept small in size, leaving a maximum amount of memory for user programs -- A description of this new system and a user manual is in preparation and will be available in late July, 1970.

Robert Glaser
Director, LRDC

Robert Fitzhugh
Director, Computer Support Facility

May, 1970

TABLE OF CONTENTS

Foreword iii

Introduction 1

System Hardware 5

    Main Frame 5

    Input/Output Devices 15

    Subject Terminals 17

System Software 26

    Central Executive 26

    Operator Control and Error Detection 41

    Peripheral Equipment Control 44

    Utility Programs 58

    Higher Level Languages 60

Documentation 63

Physical Plant 65

Personnel 67

Experimental Control Programs 72

    Angle Discrimination 73

    Sum and Recall 75

    PALL I 77

    PALL II, III 79

    PALL IV 81

TAP II 83

Letter Discrimination 86

Preferences 88

Digit Memory Span 91

Object Memory Span 92

Logical Classification and Concepts of Relationship 93

Simple Spelling 98

On Developing a Computer Laboratory 100

Installation and Development 102

Personnel and Management 107

References 114

Appendix A: Index of Computer Facility Documentation 120

Appendix B: Outline for the Documentation of Systems Programs 125

THE DEVELOPMENT OF AN ON-LINE LABORATORY FOR CAI AND BEHAVIORAL RESEARCH (1964-1968)

Wilson A. Judd

Learning Research and Development Center

University of Pittsburgh

INTRODUCTION

Research in the area of Computer-Assisted Instruction (CAI) has been and will continue to be one of the major programs of research of the Learning Research and Development Center. Over the past four years, the Center has developed a Computer Facility to support this program. This Facility has had two major objectives: (1) to provide the equipment and services necessary to support the research and development efforts of the Center's experimental psychologists and others working in the area of the educational technology, and (2) to conduct a research and development program in the area of CAI systems and terminal design. The interaction of these two objectives and the research orientation of the Center staff has resulted in the creation of a Computer Facility that might be viewed as lying midway between a CAI installation and a computer-based laboratory for behavioral science research.

Since one purpose of the CAI program was to explore the characteristics of a desirable CAI system, the system itself was built up from basic components rather than being installed as a complete unit. The characteristics of student terminals suitable for CAI was a matter of particular interest (Glaser, Ramage, and Lipson, 1964), and consequently, a very flexible system, incorporating a variety of terminal devices, was developed. The LRDC Facility is quite small in comparison with most existing CAI installations, and the system itself was not intended to support a full-scale CAI production effort. Rather, the emphasis has been on supporting the development of small-scope CAI programs designed for the purpose of evaluating specific instructional strategies and investigating the behavioral characteristics of students in a CAI environment. In addition, the Facility has proven to be quite useful for the control of psychological experimentation, some but not all of which has derived from questions raised by the CAI development effort. Since it appears that the emphasis of the LRDC CAI program may shift toward CAI production on a broader scale in the near future and because of the author's particular orientation, this report will view and attempt to describe the LRDC Computer-Facility as a computer-based psychological laboratory with a particular orientation toward research problems related to CAI.

How can a computer system such as the LRDC Facility be used in behavioral experimentation? First, the system can be used for stimulus presentation. Alpha-numeric and some limited graphic materials can be presented by means of cathode-ray tube (CRT) displays. More complex visual displays can be presented by slide projectors under computer control. Audio messages can be presented by means of "randomly accessible" tape recordings. Any other display devices that can be controlled by a limited amount of relay logic can be computer-controlled with a minimum amount of effort. The advantages of computer-controlled stimulus presentation are very accurate timing and control and, when it is desirable, the capacity for complex, response-dependent sequencing of stimuli. Under some conditions, the stimuli are also easily modified by the experimenter or they may be generated algorithmically.

Secondly, the system may be used advantageously for response processing. In addition to the usual multiple-choice pushbuttons used in mechanized human experimentation, the subject can make constructed responses by means of a typewriter keyboard. The use of photographic stimuli projected onto a touch-sensitive surface developed at LRDC has proven to be valuable for working with young children. Again, the system provides very accurate response timing, and the ability to handle high response rates and to process complex responses such as typed words and sentences.

A third area of interest is the system's capacity for record keeping and data display. Response data stored in the system may be permanently recorded on paper tape, magnetic tape and punched cards--all record forms that are amenable to computerized statistical treatment without additional manual operations. The system itself is not suitable for running extensive statistical programs, but it is quite capable of limited data reduction and summation and can provide the experimenter with displays of his data very soon after the completion of an experimental run. It would be feasible to provide the experimenter with a running summary of the data on a teletype or CRT while the experiment is in progress, although this feature has not been utilized.

A most interesting aspect of computer control is the ability of the system to make rapid, sophisticated decisions concerning stimulus presentation based on the pattern of subject responses. Reinforcement can be made contingent on a complex pattern of responses that might be scattered over a long period of time. In paired-associate learning and concept formation tasks, lists can be manipulated and items can be introduced, dropped and re-introduced as a function of algorithms determined by the experimenter and by the subject's sequence of responses. When children are used as subjects, it has been found useful to note, but otherwise ignore, certain irrelevant responses and to repeat pertinent instructions whenever they are required throughout the task.

It is hoped that this report may prove to be of value to psychologists and educational researchers who are interested in the promises and problems of on-line control for behavioral experimentation. The hardware and software aspects of the system itself are described in detail; the intention is to present one example of an on-line control facility. While the system has proven to be relatively satisfactory, it is far from ideal. Were we to begin designing another system today, substantial changes would be made. Hopefully, the system description, as well as the sections concerning documentation procedures, the physical plant, and facility personnel will provide the interested experimenter with some feeling for the various aspects and scope of the design, construction, and maintenance of such a system.

A number of experiments which have been run under the control of the system are described in hopes of demonstrating some of the advantages of on-line experimental control and, perhaps of stimulating the reader's interest in the possibilities of "contingent experimentation." Finally, an attempt has been made to summarize some of the knowledge, experience, and folklore generated by four years of constructing, using, and living with a laboratory computer system.

System Hardware

Main Frame

All laboratory equipment control is provided by two Digital Equipment Corporation computers, a PDP-7 and a PDP-9. These two machines are highly program compatible, differing only in speed (cycle time is 1.75 microseconds for the PDP-7 and 1.0 microseconds for the PDP-9), the use of the memory-indexing registers, and some minor input/output (I/O) conventions. The PDP-9 has been interfaced to the PDP-7 in such a way that it can be used independently or as additional core storage for the PDP-7. The interface between the two machines is discussed in detail below.

The PDP-7, the first machine installed in the laboratory, can be viewed as the basic control device. As such, it has a relatively large number of options and I/O features. Some of these were installed by Digital Equipment Corporation at the time of purchase. Others have been added by the LRDC staff as required by the expansion and development of the system. Core memory in the PDP-7 consists of 16K (K = 1024) or 16,384 eighteen-bit words, divided into two 8K fields. Normally, a program may address only those locations within its own field. Data transfers and program jumps across field boundaries are accomplished by directly addressed instructions while in the memory extension mode.

It would seem worthwhile to pause here for further explanation. The instruction LAC X will load the contents of location X into the acculator (AC). The instruction LAC I X (load indirect X) will use the contents of location X as the effective address from which to load. Thus, if location X contains the number Y, the instruction LAC I X will load the contents of location Y into the AC. Likewise, again assuming that location X contains the number Y, the jump instruction JMP I X will cause program control to be transferred to location Y. If the machine is not in the extended memory mode when an indirect instruction (or any instruction, for that matter) is executed, only the last 13 bits of the address are considered. Since $2^{13} = 8,192$, this is sufficient to address any location in a single 8K field. If the extended memory mode is in effect, the address is treated as consisting of 15 bits but only for an indirectly addressed instruction. Thus, a total of 32,768 locations are accessible under this addressing structure, but some degree of protection between fields is provided by the requirement that transfers across field boundaries take place only by indirect addressing while in the memory extension mode. Memory extension mode itself is under program control.

A standard PDP-7 feature related to indirect addressing is the autoindexers. Locations 10 through 17 in each of the two 8K fields in the PDP-7 are specified as autoindexers. When these locations are addressed by normal, i.e., not indirect, instructions, they behave as any other location. However, when they are addressed by an indirect instruction, e.g., LAC I 10, the contents of the autoindexer word are incremented by one and the new value is used as the effective address for the instruction. Thus a loop containing a LAC I 10 instruction, where location 10 contains the value 100, would systematically load the contents of successive locations starting with location 101.

One vendor-supplied option is the Extended Arithmetic Element (EAE). This consists of an 18-bit multiplier quotient register (MQ), a six-bit step counter register (SC), two sign registers, and the EAE control logic. The EAE's purpose, in addition to providing an additional register, the MQ, for data transfer between routines is to facilitate high-speed multiplication, division, shifting, and bit manipulation.

Since the primary purpose of a computer in a behavioral science laboratory is equipment control rather than numerical processing per se, its I/O capabilities are of prime importance. The PDP-7, being designed as a process control machine, is fairly well suited for such an application. All system I/O is handled by some combination of the following devices: Information Collector, Information Distributor, Device Selector, Automatic Priority Interrupt System and Multiplexor.

Information transmission and device control. The Information Collector (IC) reads data from a specified buffer into the AC or MQ whenever a particular input-output transfer (I/OT) instruction is executed. Likewise, the Information Distributor (ID) loads a specified buffer with the contents of the AC when a particular I/OT instruction is executed. The Device Selector (DS) decodes each I/OT to determine which device buffer should be loaded into or from the AC. Six bits in each I/OT instruction are used to specify the type of I/O device. This means that the I/OT structure can handle up to 2^6 or 64 different types of devices. If there is only one device of a particular type, then only one buffer is associated with this device type. In some instances, e.g., the case of subject terminals, there are several I/O devices of a particular type, each with its own input and output buffer. For this purpose, there are four sub-device selection bits available in each I/OT instruction. Thus, up to 16 input or output devices of any one type can be handled by the I/OT structure. Currently, the LRDC system has 28 input device channels to the AC and seven device channels to the MQ. Data transmitted to all subject terminal displays, to the paper tape punch, and to the teletype printers are transmitted via the accumulator and the Information Distributor. The Information Collector handles all data from subject terminal inputs, teletype keyboards, the card reader, and the paper tape reader.

Automatic Priority Interrupt System. Since the peripheral I/O devices are, as a rule, much slower than the Central Processing Unit (CPU), it would be extremely inefficient to make the CPU wait until the peripheral device is ready to transmit or receive its next piece of information. This is particularly true if the data input rate is dependent on human response times, as is the case of the computer used in the behavioral science laboratory. This mismatch in processing speed is the basis for the time-sharing system. While the system is waiting for input from one device, it can proceed with other activities such as numerical processing, or transmitting data to or from another I/O device. There must be some means, however, of notifying the CPU when a device is ready to transmit or receive data. For the PDP-7, this function is fulfilled by Automatic Priority Interrupt System (API).

When an I/O device is ready to transmit or receive data, it activates its assigned API channel. This action interrupts the ongoing program by causing program control to be transferred to a specific location in core. In the PDP-7, there are 16_{10} API channels for which the corresponding core memory locations are 40_8 to 57_8. Instructions stored in these locations then cause control to be transferred to a routine appropriate for controlling the relevant I/O device. When the equipment control routine has completed its necessary function, program control is returned to the original program by the software time-sharing system. It is obvious that a problem would arise if two or more interrupts were allowed to occur simultaneously. This is prevented by the interrupt priority scheme. All the interrupt channels are ranked on a priority basis with channel O (location 40_8) having the highest priority and channel 17_8 (location 57_8) having the lowest priority. When a "break" occurs on a particular channel, that is, when that channel is activated by an I/O device, all lower channels are, in effect, turned off. The interrupts are not lost, but they are prevented from interrupting the ongoing program until a "debreak" command is issued. Breaks on higher priority channels are allowed to interrupt routine processing the previous interrupt. If a particular type of I/O device requires immediate service or if it is of crucial importance to the operation of the system, it is assigned to a high priority channel. Devices which can tolerate some delay in their rate of data transfer are assigned to relatively low priority channels.

The current API channel device assignments are given below. The different devices mentioned will be discussed in greater detail in subsequent sections.

Channel 0 is assigned to the Control Teletype. While the Control Teletype does not require fast service for its data transfers, it is desirable that it be the device least susceptible to being blocked out of the API so that it can be used to restart the system in case of a crash.

Channel 1 is currently not in use but is tentatively reserved for the use of an eye-movement camera, a device with an extremely high data rate.

Channel 2 services the Touch Sensitive Surfaces, a set of subject response devices which have no external buffering and consequently require very fast service.

Channel 3 is assigned to a two-millisecond clock used for placing time limits on subject responses.

Channel 4 is currently unassigned.

Channel 5 is used to detect status signals from the magnetic disc such as "ready to start data transmission" and "data transmission complete".

Channel 6 is used to detect similar status signals from the PDP-9 computer.

Channel 7 detects status signals generated by the magnetic tape controller.

Channel 10_8 services a variety of devices falling in the general category of in-house I/O such as the Paper Tape Reader and Punch, Card Reader, and in-house teletypes.

Channel 11_8 is used to detect the presence of status signals from the Dataphones, used to control the operation of remote teletypes over telephone lines.

Channel 12_8 services the random-access audio units. These are relatively slow devices and can, therefore, tolerate relatively long data transmission delays.

Channel 13_8 is currently unassigned.

Channel 14_8 services the keyboards used for subject responses. Data from a keyboard is saved in an external buffer and is not destroyed until the keyboard is reset by the control program. Therefore, these devices can tolerate indefinitely long delays before transmitting data.

Channel 15_8 is currently unassigned.

Channel 16_8 detects status word transmissions from the Random Access Slide Projectors. Again, the data to be transmitted can be saved indefinitely in an external buffer.

Channel 17_8 is used by the Time-Sharing System software. Since Channel 17 has the lowest priority, an interrupt on this channel will not be recognized until all interrupts on the higher priority channels have been serviced. The system software uses this relationship to determine the time at which normal processing should be resumed.

Multiplexor. While the I C and I D provide convenient and flexible means of transmitting data to and from a variety of devices, the requirement that data first be loaded into the A C or M Q is restrictive and time-consuming when large amounts of data must be transmitted. A multiplexor allows bulk data to be transmitted directly between an I / O device and core memory without passing through the A C. Control commands to the I / O device and information about the data, e.g., the core address of the data or the amount of data to be transferred, are transmitted via the A C and the I C or I D. The data transfer, itself, is done on a cycle-stealing basis. That is, while the CPU continues to run an ongoing program, some cycles are used for transferring data. The "stolen" cycles have no effect on the ongoing program other than its running at less than its normal speed.

The multiplexor consists of eight bi-directional channels. Four of these are currently implemented servicing the disc, the magnetic tape controller, the line printer, and a clock. Normally, the multiplexor unit checks one channel during each machine cycle on a round-robin basis. When an I / O device is ready to transmit or receive data, it issues a data break request on its assigned channel. When that channel is queried, one eighteen-bit word is transferred to or from core. Since all eight channels must be monitored, the maximum data transfer rate for any one channel is 71, 000 words per second. The magnetic disc, however, requires a transfer rate of about 108, 000 words per second. Currently, this problem is overcome by locking out the other seven channels while a disc transfer is in progress. In the near future, a more sophisticated priority system will be installed which eliminates the undesirable lockout procedure.

Timing. Accurate timing is of considerable importance in a laboratory computer system. In general, the user has two types of interests in timing during an experiment. First, he may wish to control the length of time that the stimulus is displayed, the inter-item interval, or the time allowed for the subject to respond. Secondly, he may wish to measure response latencies. The L R D C System has two clocks to serve these functions. The first is a two-millisecond clock for controlling time delays. Although the clock is accurate to ± 2 msec., time delays are requested in 10 msec. units as a matter of convenience. This is essentially an alarm clock. For example, a program might display a stimulus and then request that it be notified when a certain length of time has expired. The clock is set to a negative number corresponding to the length of the time delay. When the clock count reaches zero, it causes an interrupt on A P I Channel 3. The system software then determines which program is waiting for that particular time delay and notifies it that the delay has expired.

A second, 1 msec., clock is used to measure response latencies. This clock, called the Time-of-Day Clock, is an external 36-bit buffer which is simply incremented once every msec. I / O T instructions enable the user to read the contents of the clock buffer into the A C and M Q. In practice, the clock is automatically read by all of the system routines which control devices at the subject terminals. This assures that the clock reading is taken within a msec. of the time at which the stimulus display was completed or at which the subject made his response. These times are saved and are available to the user program on request.

Additional device control features. Some subject terminal devices such as the teletypes and keyboards are controlled directly by the I / O T instructions while others are driven by relay logic. The P D P -7 has four general purpose relay buffers, each containing 18 bits or 18 single-pole, single throw relays. A system relay control routine allows a user program to set or reset any particular relay or group of relays. Since the relays are mechanical devices, they are relatively slow, requiring about 5 msec. to settle. Consequently, the relays themselves are systematically being replaced with solid state logic. Currently, two buffers are used to control the Random Access Audio units. A new audio system will be controlled by a separate set of solid-state buffers. Fifteen relays (or their solid state equivalents) are employed in the control of the slide projectors and cathode-ray tube displays. The remaining 21 relays, 55 following the installation of the new audio system, are available for controlling special devices, such as lights or buzzers, requested by the experimenters.

Finally, a two-channel digital-to-analog (D/ A) converter is used for vector control on the CRT displays, one channel for the X-coordinate and the other for the Y-coordinate. Both D / A channels can be switched rapidly from one CRT to another via the solid state logic discussed above. This allows different displays to be presented on several CRT's without apparent delays. The system is also equipped with an A / D or analog-to-digital converter but thus far, no experiments have been requested which would require this feature.

The PDP-7 / PDP-9 interface. During the laboratory's third year of operation, it became obvious that an additional 16K of core memory would be advisable. Due to a series of difficulties and delays in the ordering and delivery of the additional core, the possibility of using a 16K PDP-9 computer to fulfill the extra core requirements was investigated. The PDP-9 was a newer Digital Equipment Corporation product, highly compatible with the PDP-7 and essentially a newer model of it. Analysis of the problem of interfacing the two machines indicated that such a solution to the core problem was feasible. Due to advances in the state-of-the-art of computer construction, it was possible to purchase a stripped-down version of a PDP-9 for $11,000 less than the cost of 16K of additional core for the PDP-7. When the cost of interfacing the two machines is taken into account, the net savings were $6,000.

The PDP-7 to -9 interface has three modes of operation which are selected by a manual switch. The first is a stand-alone mode, with the PDP-7 and PDP-9 running as two separate machines. As was mentioned above, the PDP-9 was purchased as a stripped machine. Since the PDP-9 installation, an EAE unit has been constructed and installed in the -9, thus increasing the number of instructions to a set that is essentially equivalent to that of the PDP-7. Although it is relatively rare, situations do arise in which it is economical to run different programs on each of the two machines.

The second, and most commonly used, mode is the extended-core mode. In this case, the PDP-9's CPU is idle and its memory is used as the upper 16K of the PDP-7. In this mode, the PDP-7 has a cycle time of 1.75 microseconds when accessing its own core and 2.25 microseconds when accessing the PDP-9's core. This is the mode of operation for the time-sharing system. As the system is presently designed, the system itself, the Executive, resides in the PDP-7 while user programs being run reside in the PDP-9.

The third mode of operation is the multiple-processor mode in which the CPU's of both machines are active, the PDP-9 in its memory and the PDP-7 in both its own and the -9's memory. It will be recalled that while the PDP-7 has a basic cycle time of 1.75 microseconds, the newer PDP-9 has a 1.00-microsecond cycle time. The difference between these cycle times is accommodated by splitting the read-write memory cycle during a PDP-7 access in PDP-9 core. Consequently, the -9 loses 1.6 microseconds when the PDP-7 accesses -9 core. The PDP-7 operates with a 1.75-microsecond cycle time when accessing its own core and a 3.65-microsecond cycle time when accessing the

PDP-9 memory. This mode is used primarily for running PDP-9 vendor-supplied software, such as FORTRAN and the PDP-9 MACRO Assembler, which requires the use of I/O equipment controlled by the PDP-7.

Input-Output Devices

The 32K words of core storage described above are supplemented by a 640,000 word magnetic disc. This is a Burroughs model 9370 High Speed System memory with fixed heads and 100 tracks on each side of the disc. Each track is divided into 100 segments and each segment is divided into 100 eight-bit bytes. The disc is addressed in binary coded decimal and in either byte or word mode. A single segment can contain 50_{10} PDP-7 words in byte mode or 32_{10} PDP-7 words in word mode. The transfer rate, through the multiplexor, is 310,000 bytes per second or 108,000 words per second. Maximal access time is 34 msec.

Bulk storage is provided by a single Datamec D2020 tape drive. This is a 7-channel unit which can read and write IBM compatible tape at either 200 or 556 bits per inch. Tape speed is 45 inches per second. The increased use of the laboratory facility has demonstrated that this single drive is not sufficient for the requirements of the time-shared system. During the second half of 1969, the current tape drive will be replaced by four Texas Instruments Model 959 single capstan tape transports which will read and write a 556 or 800 bits per inch with a tape speed of 120 inches per second.

In addition to magnetic tape, input to the system is provided by punched cards and punched paper tape. The card reader is a pneumatic pick, photoelectric-read unit manufactured by General Design, Inc. The maximum read rate is 600 cards per minute. Currently, data input is via the AC, but direct memory access capability will be provided shortly by interfacing the reader via the multiplexor unit.

Both the PDP-7 and the PDP-9 are equipped with paper tape readers. These are photoelectric-read devices which sense holes punched in five-, seven-, and eight-channel punched paper tape. The standard input medium is 8-channel, fan-fold paper tape, read at the rate of 300 lines per second. Information can be read from the tape in either alpha-numeric or binary mode. In alpha-numeric mode, each eight-bit line of tape corresponds to one ASCII (American Standard Code for Information Interchange) character. In binary mode, six-bits from each of the three successive lines are read to compose one 18-bit binary word. Data transfer is via the AC.

Data output from the system is recorded by punched paper tape and a line printer. In some cases, data are recorded on magnetic tape for subsequent statistical processing by the University IBM 7090 or 360 computer systems. It is anticipated that a card punch will be installed during late 1969. Both the PDP-7 and PDP-9 have paper tape punches which punch either ASCII or binary code at the rate of about 63 lines per second. Data transfer to the punch is via the AC. Although paper tape is a somewhat difficult material to work with (tapes tend to be too large and wear out rapidly with use) the reliability and flexibility of the paper tape reader and punch has made paper tape the system's most popular medium for both programs and data. It is anticipated that the use of paper tape will decrease with the installation of the expanded magnetic tape system and the card punch.

Hard copy listings of programs and data is provided by a Porter HSP-3502 Medium Speed Chain Printer. This device has a character set of 96 characters (numerals, upper and lower case letters, and special characters) and prints at the rate of 200 lines per minute, each line consisting of 132 character positions. Data are supplied to the printer from the CPU via the multiplexor unit.

Operator interaction with the system is provided by the console switches on the PDP-7 and PDP-9 and by three teletypes. The PDP-9 has one console teletype, used for the control and debugging of programs, usually batch or utility function programs, run on the PDP-9. The PDP-7 console teletype is the Control Teletype mentioned earlier which is assigned to API Channel Zero. An extensive software package allows the operator to control the time-sharing system by means of this device. The second PDP-7 teletype is treated as a standard I/O device and is assigned to API Channel 10. This teletype is primarily for program debugging purposes. It is normally used by a programmer to control a debugging utility program. Alternatively, this device can be treated as a subject terminal teletype, thus allowing the programmer to check the operation of his program without leaving the computer room. All teletypes transmit data to and from the CPU via the AC.

Subject Terminals

As was previously discussed, one major emphasis of the LRDC CAI program has been the exploration of the student/subject matter interface and the development of student station devices for CAI. (Glaser, Ramage, & Lipson, 1964; Glaser & Ramage, 1967) Consequently, a fairly wide variety of subject terminals is available to the experimenter.

Teletypes. The simplest terminal is a standard KSR-33 Teletype. Currently, one teletype is in use as a student terminal and a second is being used as a teacher terminal in conjunction with another student station. These devices are additional to the previously discussed console and debugging teletypes in the computer room itself. Ten more teletypes are on order and will be installed during the spring of 1970. As a subject terminal, the teletype is noisy, fairly inflexible in its format, and slow, printing at a maximum rate of ten characters per second.

It does, however, have the advantage of being quite reliable and relatively inexpensive.

Many of the disadvantages of the standard teletypes can be alleviated by replacing the teleprinter unit with a cathode-ray tube (CRT). The LRDC laboratory has two such terminals composed of teletype keyboards and Tektronix CRTs. While these two devices are usually used together in a single terminal, either may be used separately as a component of some other station.

Keyboards. The keyboards currently in use are standard teletype keyboards which have been modified slightly so as to require an unlock command from the computer and to provide a parity bit. Characters typed by the subject are transmitted to the computer one character at a time. The keyboard is automatically locked following each key strike and must be unlocked by a command from the computer. A light on the keyboard in parallel with the unlocking mechanism indicates when the keyboard is activated. Such a procedure might be considered to place an unnecessarily heavy load on the computer system, but it does allow the experimenter's program to maintain very close control over the subject's response behavior. With the system loadings experienced to date, the time required to unlock the keyboard is not noticeable. There are, however, some disadvantages to the current keyboards. Due to their modification, the keyboards produce a non-standard code, and this is now undesirable since all other devices in the system use the standard ASCII code. The keyboards are not as reliable as might be desired and are noiser than newer keyboards. The keyboards of several different manufacturers are currently being evaluated, and it is anticipated that the teletype keyboards will be replaced shortly.

Cathode-ray tube displays. The original CRT displays installed were Tektronix RM-564 Storage Oscilloscopes. These devices, which are rather bulky, have a three-by-four inch screen coated with a phospher which will retain or store an image for several minutes without noticeable deterioration. Thus, the computer program can display a stimulus on the screen once and have it persist for as long as it is required without the necessity of the display being constantly refreshed. When the CRT is used in this fashion, in storage mode, the display must be static; it is not possible to create a stored moving display. Additional information can be added to a stored display, but the display cannot be selectively erased. The entire screen must be erased at once. Erasure requires approximately one-fourth of a second.

The CRT can also be used in dynamic mode. In this case, the display is not stored and must be constantly refreshed. This allows the user to present a moving display and to erase or replace selectively individual portions of the display. The display itself is more legible than a storage mode display since there is a fairly high level of background illumination on the screen in storage mode. The problem with dynamic mode displays, of course, is that a substantial amount of CPU time is required to constantly refresh the display. Consequently, storage mode is much more widely used than dynamic mode, although dynamic mode is available for situations which require highly interactive displays. Three such CRTs are currently in use, one in a standard student station and two in subject terminals constructed for a specific experiment.

In a second standard student station, the RM 564 'scope' has been replaced by a Tektronix 601 Storage Display Unit. This unit, which was designed specifically as a terminal display, has all the features described above but does not have several other features provided by a general purpose oscilloscope such as the RM 564. Consequently, the 601 display unit is less than half the overall size of the RM 564 'scope'. Both devices have the same screen size, but the 601 unit presents a more legible display since the background illumination is less variable than that of the RM 564.

A new student station under construction will incorporate yet a third type of CRT display, a Tektronix 611 Storage Display Unit. This device has a seven-by-nine inch display screen which incorporates a "write-through" feature. That is, a dynamic display can be superimposed over a stored display. Thus, a complex display might be stored on the CRT while a movable cursor indicates the location of the next character to be typed. It is anticipated that the RM 564 displays currently being used in student terminals will be replaced by 611 Display Units during late 1969.

All of the above units are driven by the same CRT control routine. Since the system does not include a character generator, alphanumeric characters are formed by the software by selecting the appropriate points to form the requested character from a five-by-seven point matrix. This routine provides an experimental control program with upper and lower case letters, the ten digits, subscripting capabilities and a limited set of special characters. An experimental control program may also define its own character set, such as the Cyrillic alphabet, by providing a set of tables which define the appropriate points to be selected from the basic 35-point matrix. A point-plot option in the CRT control program allows the experimenter to create graphic displays beyond the limits of the 35-point matrix. This operation is quite tedious, however, and has not been used extensively. A line drawing program has been developed for the 611 Display Unit and may be incorporated as a routine of the System CRT control program when more of the 611 Units are operational.

Slide projectors. While the CRT units can provide fast and flexible stimulus displays, the complexity of the displays is quite limited. Current CRT displays are incapable of presenting half-tone or multicolored displays, and only the simplest line drawings are feasible. All of these limitations can be overcome by displays generated from film. The LRDC System uses 35 mm. slide projectors for this purpose. Filmstrip projectors were rejected because of the difficulty of altering the sequence of frames in the filmstrip.

The original projectors used were Kodak AV 900 projectors. These devices were not designed for random access use and required one second to skip each slide. Thus, if a subject's response to slide 10 required that slide 20 be shown next, there was a ten-second delay before slide 20 was presented. The current projectors are the more recently developed Kodak RA-950 devices as modified by Mast Development Corp. The RA-950 can access any of its 80 slides within approximately 4 seconds. The minimum access time is still just under one second. The only LRDC modification was to place the shutter under program control. The projectors are used in pairs allowing a second slide to be superimposed on the first for purposes of providing knowledge of results, etc.

Touch-sensitive surfaces. While the projectors can be incorporated into any experimental terminal, their most common use is in conjunction with a Touch-Sensitive Surface. This is a translucent screen composed of a matrix of square touch-sensitive elements. When a subject responds by touching some component of the slide projector display, one of the touch-sensitive elements is activated and the program is able to determine which area on the surface the subject touched and, consequently, which aspect of the display was touched. The original display, developed jointly by Westinghouse Electric Corporation and LRDC, consists of a heavy sheet of plexiglas 18 inches square. This surface is divided into a nine-by-nine matrix of two-inch squares, the squares being separated by narrow ridges raised one-eighth of an inch above the surface. These ridges hold a network of fine wires, spaced one every third of an inch, just above the surface. The horizontal and vertical wires are separated by 1/32 of an inch. The five wires corresponding to a particular row or column of the matrix are all tied together at the edge of the surface. A sheet of soft, flexible plastic covers the entire grid. When a subject presses anywhere on one of the squares, one or more of the horizontal wires is pressed down onto one or more of the vertical wires. In effect, this completes a switch closure which uniquely defines the square lying at the junction of row and column activated.

This device, which was designed for use with young children, has proven to be quite successful. Two identical surfaces, each with a pair of RA - 950 projectors, are currently in operation. There are some drawbacks to the design discussed above. The ridges between the squares create dead areas and, to some extent, interfere with the slide projector display. There is a slight paralax problem since the slide image is actually projected on the rear of the plexiglas surface. The surface itself is larger than necessary and for most applications it is masked off so that only the center 25 elements are exposed to the subject.

The development of an improved Touch-Sensitive Surface has been a major project of the LRDC engineering staff. Several different approaches to the problem are currently in progress. A smaller screen, based on the same wire to wire contact principle, has been constructed and will be evaluated in conjunction with a new projector. Other alternatives under development will be discussed in a subsequent report (Fitzhugh & Katsuki, in preparation).

Random-access audio. Since much of the laboratory's work is conducted with small children, it is essential that some type of audio presentation be available. Random access to audio is one of the most pressing problems in the area of CAI terminals. Most of the units available commercially use multi-channel magnetic tape recorders which are, for the most part, far too slow for response-contingent audio presentations. While there are some very fast access audio units available, these are still very expensive. While the audio units employed by the LRDC laboratory have some serious deficiencies, they appear to be one of the more feasible alternatives, given the current state-of-the-art.

These units are the Westinghouse-designed CROWs (Computer Random Oriented Words). The recording medium is a six-inch-wide dictaphone belt containing 128 tracks. These tracks are accessed by a bar containing 16 record/play heads. The bar itself may be placed in any of eight different positions. The belt moves at the rate of three and three-quarter inches per second. Thirty inch belts, containing eight seconds of audio on each track, can provide up to 17 minutes of audio messages. A photoelectric circuit detects holes punched in the edge of the belt and divides each track into eight one-second segments. The minimum addressable message length is, therefore, one second. Since a single message may be continued from one track to another, the maximum message length is essentially 17 minutes. The units were designed to have a rapid reverse feature, but this was found to be unreliable and is no longer employed by the LRDC System. As a result, the CROWs can have almost instant access to 128 messages, one of each track, and access to all message units on the belt within a maximum of seven seconds. While faster access would be desirable, it has been our experience that delays may be kept to a tolerable level by the strategic placement of redundant messages on each belt.

Two CROWs are installed in the system. For the most part, one of the units has been available for use with subjects while the other has been used by the engineering staff to install and test modifications to improve the fidelity and reliability of the units. Currently, a battery of six units, all modified, is being installed. Although the CROWs were designed so that each unit could be used by several subjects, the availability of a number of units makes it possible to assign a different unit to each subject on the system. This keeps the access time within reasonable limits and has greatly reduced the required size and complexity of the control routine.

Recording a CROW belt is a time-consuming process. To alleviate the demand for computer time, a Sony tape recorder was modified to provide an off-line method of making recordings. The tape recorder simulates the eight sector, 128 track format of the CROW belt. The user records his messages in the desired sectors on the recorder's linear tape. At a later time, the tape recorder is coupled to the system and the recording is transferred to a CROW belt in 17 minutes. Subsequent editing of the belt is done on-line under the control of a special-purpose CROW recording program.

Terminal flexibility. It should be emphasized that all of the devices discussed thus far may be used in any combination. Although the CRT and keyboard are used together in two of the student stations, there is no reason why a keyboard could not be used in conjunction with a slide projector. CRT displays are used in conjunction with a number of different response devices, and audio can be included as a supplement to any subject terminal. The flexibility of the system is demonstrated by the variety of special-purpose terminals constructed by various experimenters. The Touch-Sensitive Surface appears to the CPU as a set of 81 momentary contact switches. Therefore, any response device which uses momentary contact switches can serve as input to the Touch Surface control routine. Any display which can be controlled by a reasonable number of relays can be driven by the system with a minimum of effort. One experiment, investigating response latencies, has used a number of specially designed pushbutton keyboards in combination with a C R T display. Another, concerned with discrimination learning in very young children, has used a slide projector and a single element touch sensitive screen. A variety of bells, lights, buzzers, and M & M dispensers have been attached to terminals for specific experiments.

Remote terminals. In addition to in-house experimentation, the system has the capability of conducting remote operations. Two dataphone channels and two recorder coupler channels allow the operation of terminals consisting of teletypes and audio at any location at which two telephone lines are available. Program input to the dataphone control routines is identical to the input required by the in-house teletype routines. Connecting the CROW audio units to the recorder coupler is a simple patchboard operation. Therefore, programs which can be run on local terminals consisting of teletype and audio can also be run at a remote location.

System Software

Central Executive

The original design of the LRDC time-sharing system was provided by Arthur Kaupe (1966) of Westinghouse Electric Corporation. The initial programming of the scheduling and memory management routines was done under his supervision at the Westinghouse Research Laboratories and was documented by Bright (1965).

Jobs and job scheduling. The basic operational unit in the LRDC time-sharing system is a "job." In the most general instance, a job corresponds to the line of code controlling the terminal being used by a single student or subject. In this case, the subject is assigned a particular job number when he is signed onto the system and is identified by this job number throughout his use of the system for that particular session. In other instances, a job may correspond to a data-reduction program being run on the system or may even consist of a short equipment-control routine called by the system for a particular function which cannot be handled under the auspices of a subject's job. The critical aspect of the job concept is that only one job can be running at any one time. All other jobs in the system are either queued, waiting for an opportunity to run, or are suspended, waiting for the occurrence of some particular event.

Let us first consider the conditions under which a job is suspended. Since experimental control and CAI programs require relatively little processing time as compared with the time required for a subject to respond, a job is usually allowed to complete the processing necessary to evaluate and store the data from the subject's last response and to present the next stimulus. A job is never allowed to run while waiting for a subject's response. In all, there are six conditions under which a job may be suspended. Suspension is mandatory under the first two of these conditions. (1) Whenever a job requests a subject response (by activating the subject's response device), it is automatically suspended by the device control routine. The job will not be set up, that is, made ready-to-run, until the subject makes a response. (2) A job may request a time delay, that is, it may request that it be notified when a certain period of time has expired. As in the case of a response request, the time-delay routine automatically suspends the job until the expiration of the time delay. Time delays are often requested in conjunction with some other event. For example, it might be desirable to place a time limit on the subject's response. In this case, the job requests a subject response or a time delay in its call on the response device control routine. The job is then automatically suspended for both of these reasons and will not be set up until a subject response is made or until the requested time delay has expired.

(3) The third suspension condition is voluntary. Some of the stimulus display devices, such as the slide projectors and random-access audio units require relatively long periods of time to present the requested stimulus. Consequently, the control routines for these devices provide options for suspending the job requesting the stimulus. Normally, a job will ask to be suspended until the requested stimulus has been displayed but in some instances a job may wish to start the stimulus device moving toward the requested stimulus and then do additional processing or request additional displays from a second stimulus display device. For example, a job might request that a particular slide be positioned but not shown and then, while the slide is being positioned, the job might request that an audio message be played. Similar options exist in the I/O transfer routines. If a job records data from subject responses in one specific buffer and then requests that the filled buffers be written out onto disc, it would request suspension until the disc transfer is completed so as to insure that all of

the relevant data have been copied onto disc before new data is written into the buffer. If, on the other hand, a job records data from successive responses in a series of successive buffers, a second buffer may be being filled before the first has been copied. In this case, a job could request disc transfers without suspensions and thereby avoid any delays due to waiting for the completion of the disc transfer.

(4) The fourth suspension condition is required as a result of the suspension options discussed above. Assume that a job requests that a slide be positioned and does not request that it be suspended so that it is free to do further processing while the slide is being positioned. The job must now insure that the slide is in position before making other requests such as a student response request. Under such conditions, the job may request a suspension, via System Routine WAIT (Judd, 1967), until the completion of a previously requested event, in this case, the positioning of a slide. The job will then be suspended until the slide is in position. If the condition has already been satisfied at the time that the request is made, job simply continues in the ready-to-run state.

(5) A job may also be suspended by the system itself. If the system detects an error in a job's call on the system or an equipment malfunction, the job is suspended by system routine SYSERR (Buckwalter, 1966), and can only be restarted by the system operator. (6) Finally, the system operator can suspend a job by means of the control teletype. Again, under this suspension condition, the job can be restarted only by the operator.

It is apparent that while any one job may be suspended most of the time, there will be instances in which more than one job is ready to run at a particular time. When a job is set up, that is, when the suspension conditions have been satisfied, it is queued. That is, it is placed in the ready-to-run queue. Its position in the queue is determined by its rank and the time at which it became ready-to-run. Each job in the system is assigned a rank from one to seven, with one being the highest rank. Higher-ranked jobs are always placed ahead of lower-ranked jobs in the ready-to-run queue. Thus, a job controlling an experiment and having real-time processing requirements, might be given a rank of four while a data reduction program, with no real-time constraints, might be given a rank of five. If both jobs became ready to run at the same time, the higher ranked experimental job would always be placed first in the ready-to-run queue. Likewise, if the data reduction job were running when the experimental job became ready-to-run, the higher ranked experimental job would be given control of the central processor and the lower ranked data reduction job would be pushed back into the ready-to-run queue. As a result, a higher ranked job need never wait for a lower ranked job to run and, conversely, a lower ranked job can never run if a higher ranked job is ready-to-run.

On occasion, a second experimental job is introduced into the system which has the same rank as the first experimental job. The order of jobs of equal rank in the ready-to-run queue is simply determined by which job became ready-to-run first. If a job is running when a second job of equal rank becomes ready-to-run, the first job retains control of the central processor and new job is placed in the ready-to-run queue following all other ready-to-run jobs of equal rank and ahead of any jobs of lower rank.

If, at any particular time, there are no jobs which are ready-to-run, control of the CPU reverts to System Job Zero. Job Zero has the lowest permissible rank, seven, and is the only job assigned this rank. It never suspends but can be pushed back into the ready-to-run queue by any other job which becomes ready to run. The primary purpose of Job Zero is to keep the CPU in an active state while no other jobs are ready-to-run. The minimum program required for this function is simply a short counting routine, TWIDDL, which runs in a tight loop. Any program which never requests a suspension, however, can be run as Job Zero. Under normal operating conditions, the only other program run as Job Zero is a utility routine DEBUG (Fitzhugh, 1969) which allows an operator to examine or alter any location in the system. Job Zero is switched back and forth between TWIDDL and DEBUG by means of the Control Teletype.

Master tables. It is obvious from the above discussion that relatively elaborate bookkeeping procedures are necessary to keep track of the state of the system and to preserve the information relevant to each of the various jobs. It is essential that none of the information with which a job is working be disturbed during the time that it does not have control of the CPU. If a job suspends, say for a subject response, it must be restarted at the appropriate point, with access to the response data as soon after the occurrence of the response as possible. When several jobs become ready-to-run at approximately the same time, they must be ordered so as to optimize the use of the central processor with respect to the jobs' various ranks.

The heart of the bookkeeping system is the Job Status Table (JST). When a job is introduced into the system by means of the Control Teletype, the system creates a JST for the new job and places the address of the JST in a second table, the JST Directory. This directory is simply a fixed length list of JST addresses ordered by job number. Thus the address JSTDIR + 23 (the twenty-third location in the JST Directory) contains the address of the JST for Job 23.

The JST itself consists of thirteen words, the functions of which are as follows:

1. JSTSS -- The suspend status word. This is zero if

the job is running or ready-to-run. Otherwise, it contains a number indicating why the job is suspended.

2. JSTRS -- The Request Status Word. This indicates the status of any action which the Job has requested from a peripheral device such as an I / O transfer, a stimulus display, or a subject response. It will be recalled that a job may request a suspension pending the completion of a previously requested event. If the event has already occurred when the suspension request is made, this will be indicated by the JSTRS word and the job will continue in the ready-to-run state rather than being suspended.

3. JSTRNK -- The Rank Word. This cell contains the rank assigned to the job by the operator when the job was created.

4. JSTRNL -- The Rank-Link Word. It will be recalled that jobs are ordered in a ready-to-run queue on the basis of their assigned ranks and the time at which they became ready-to-run. This cell contains the number of the job which is next, that is, after this job, in the ready-to-run queue. Thus, the ready-to-run queue is actually a linked list located in the various jobs' JST's.

5. JSTPTC -- The Pointer-to-Core Word. When a job loses control of the CPU, either involuntarily or through a suspension, a large block of core, called COMMON, containing data being used by the job is "swapped" from one location to another by the system. When the job regains control of the CPU, the block is swapped back to its original location. The JSTPTC word contains the address at which the swapped block is stored while the job is not running. COMMON is discussed in greater detail below.

6. JSTIC -- Instruction Counter Word. This cell contains the address at which the job is to be restarted when it regains control of the CPU.

7. JSTMQ -- This cell contains the value which will be in the Multiplier Quotient register when the job is restarted.

8. JSTAC -- This cell contains the value which will be in the Accumulator when the job is restarted. If the job lost control of the CPU involuntarily due to a higher-ranked job, the JSTMQ and JSTAC contain the values in the MQ and AC when the job lost control. If the job was voluntarily suspended pending some external event, the JSTMQ and JSTAC may be used to pass data to the job concerning this event.

9. JSTID -- Identification Word. This cell contains the number of the job.

10-12. JSTTD -- Time Delay Words. Three words used by the system time-delay routine to be discussed below.

13. JSTCOM -- The Common Cell. This cell contains the number of words of COMMON being used by the job.

As was mentioned previously, the system is designed to run time-shared, re-enterent code, that is, two or more subjects may be run on the same experiment at one time by a single control program. While a single program can control the general course of the experiment, the data from the various subjects must be treated separately. In the simplest case--a paced experiment in which all subjects are given the same experimental treatment regardless of their responses--the data must, at least, be recorded separately. At the other end of the continuum, one might have a situation in which subjects were given different experimental treatments, and were allowed to proceed at their own pace and in which specific stimulus presentations were contingent on the responses of the individual subjects. In either case, it is desirable that the control program have a specific location at which it can expect to find a particular type of data and that this be the data pertaining to or generated by the subject assigned to the currently running job.

This function is fulfilled by allowing each job to have access to a block of core designated as COMMON space. In the current system configuration, COMMON consists of the Autoindexers (locations 10 to 17_8) and locations 400 to 777_8 in each of the two fields available to applications programs. Whenever a job loses control of the CPU, whether it suspends or is queued, the System transfers the contents of COMMON to a block of memory obtained from the free memory held by the system and records the location of stored block in the JSTPTC cell. (This free memory, called MEMAL Space, will be discussed below.) When the job regains control of the CPU, the data swapped out is restored in COMMON before the job is restarted. Thus, an applications program being used by several different jobs to run several subjects might refer to a location called ERRORS located in COMMON space and containing the number of incorrect responses which a subject has made during the current trial. Whenever a particular job is running, location ERRORS will contain the data pertaining to the subject being controlled by that job.

Since moving the contents of a large block of core is a fairly time-consuming process, it is desirable that no more data be moved than is absolutely necessary for each job. Consequently, it is a system requirement that each program specify how much COMMON it requires before any data is stored in the COMMON area. This value is stored in the JSTCOM cell of each job using the program and only the necessary amount of core is swapped whenever that job is suspended or queued. The Autoindexers are always considered to be included in COMMON.

This is only one of many possible schemes which might be employed in the implementation of time-shared code. One alternative is to swap the entire control program. Since a control program is usually much larger than the data which it requires for its operation for short periods of time, this would be a much more time consuming process; but it would have the advantage that it would not require the separation of program and data which is a requirement of the system discussed above. A common time-sharing practice is to swap data (and/or programs) to and from disc storage rather than simply to another part of core. This has the advantage of reducing the amount of free core space required and would allow a greater number of jobs to be run simultaneously, but it would also substantially increase the time required for swapping. An earlier version of the LRDC system provided less than 100_8 words of COMMON as compared to the current 400_8 words. It was intended that this small block, which could be swapped very quickly, would contain primarily pointers to MEMAL space. Each job would obtain its own MEMAL blocks for data specific to that job. This system proved to be much too cumbersome since all data had to be accessed indirectly.

Scheduler. Having considered the bookkeeping aspects of the system, we are now faced with the question of just how the various jobs are scheduled. The Scheduler consists of three basic components: SUSPEN, which is used to suspend a currently running job; SETUP, used to make a suspended job ready-to-run and to place it in the ready-to-run queue; and the re-scheduler (RESCH) which determines which job is to have control of the CPU at any given time.

SUSPEN is called whenever the currently running job requests a suspension by (1) requesting a time delay, (2) calling a response device control routine, (3) specifying a suspension option when calling an I/0 or stimulus device control routine, or (4) calling system routine

WAIT. Involuntary suspensions of a job by the system error routine SYSERR and by the system operator also call SUSPEN. When called, SUSPEN first checks for errors in the call and then saves the contents of the A C and M Q in the JST if the caller requested that these registers be saved. The location at which the job is to be restarted is saved in the JSTIC word. The reason for the suspension, passed to SUSPEN as a parameter, is checked against the Request Status word in the JST to determine whether the suspension condition has already been met. If so, the job is queued via Routine RESCH rather than being suspended. If the reason, or one of the reasons, for suspension is a time delay, SUSPEN calls the Time Delay routine which returns to SUSPEN when the delay has been set up. Finally, SUSPEN determines which job is next in the ready-to-run queue and calls RESCH for the actual rescheduling of jobs.

System routine SETUP may be called by any of the system routines which control devices for which a job may be suspended. When such a routine determines that suspension conditions have been met for a suspended job, the data required by the job are stored in a parameter list accessible to the job, and the routine calls SETUP, passing it the number of the job and a number specifying the reason the job was originally suspended. SETUP checks the data passed for errors and uses the job's number to locate its JST. After determining that the job was actually suspended for the condition reported by the calling routine, SETUP determines whether one of the suspension reasons was a time delay. If so, SETUP calls the time delay routine to cancel the time delay for this job. It might be noted here that the System provides the capability of suspending for a time delay or some other event. The job is set up in the event of the occurrence of the requested event or the expiration of the time delay. There are no and suspension conditions available. That is, a job cannot be suspended until the occurrence of two specified events. This has not proven to be a limitation in the design of experimental programs, since the and condition can be effectively simulated by means of repeated calls on the system.

SETUP next checks the job's rank and determines where it should be placed in the ready-to-run queue. If the currently running job has a rank which is higher than or equal to that of the new job, the new job is placed in the ready-to-run queue ahead of all lower ranked jobs and SETUP returns to the calling routine. If, on the other hand, the new job has a higher rank than all other queued jobs, it is placed at the head of the ready-to-run queue and SETUP calls routine RESCH before returning to the device control routine which called it.

The means by which SETUP calls RESCH is unique in the system. Consider the situation in which several jobs become ready-to-run within a short period of time. Each of these jobs was suspended pending some event and the system is notified of the occurrence of these events by means of interrupts on the Automatic Priority Interrupt System (API) discussed previously. As each interrupt is received, control momentarily passes to the appropriate device control routine which in turn calls SETUP. It will be recalled that an interrupt on a given channel will not be recognized until all interrupts on higher level channels have been serviced. This means that an interrupt on the lowest channel, Channel 17, cannot be recognized except at a time when there are no unserviced interrupts on any of the higher channels. Each time SETUP finds that a new job has a higher rank than all other jobs in the queue, it creates an interrupt on Channel 17. Control is passed to RESCH whenever a Channel 17 interrupt is recognized, but this recognition will not occur until all higher channels have been serviced. Thus, if a series of jobs is set up in rapid succession, there may be several jobs which are momentarily the highest ranked job ready to run, but the actual rescheduling will not take place, that is, API Channel 17 will not be recognized, until all of the jobs in the flurry of interrupts have been queued according to rank.

In addition to being initiated by an interrupt on API Channel 17, it will be recalled that RESCH may be called directly by System Routine SUSPEN. If RESCH is called by an interrupt, then the currently running job is not suspended voluntarily and must be queued. RESCH, therefore, saves the values in the registers for this job. Regardless of the origin of the call on RESCH, it then obtains a block of free memory from the Systems Routine MEMAL, stores the current contents of COMMON in this block, and records the location of the block in the old job's JSTPTC word. The location at which the job is to be restarted is also saved in its JSTIC word. The location of the new job's stored COMMON is determined from its JSTPTC word, the data are restored in COMMON, and the memory block used for storage is returned to the pool of free space. Finally, the registers are loaded with the values stored in the new job's JSTAC and JSTMQ words and the job is restarted at the appropriate location.

Memory management. As was discussed above, a fairly large COMMON area is saved each time a running job is suspended or queued. Since all swapping is done within core, this means that a large area of core must be reserved as free space available for storing the swapped data. In addition, as a job runs its course, it will often require large blocks of memory for the temporary storage of input and subject-generated data. Since a job's demand for such space will vary over time, it is desirable to provide the job with this space only at times when it is needed. Such considerations lead to the development of a memory allocation package (MEMAL, Judd, 1967) to provide jobs and components of the system with variable-sized blocks of memory as they are needed.

With the current system configuration, MEMAL space is available in all four of the 8K fields. The system, located in the two lower fields, has access to space in all four fields. Applications jobs have access to only the MEMAL space in their own field.

When a system routine or a job requires additional space, it passes the number of words required to MEMAL. MEMAL then searches its list of available memory until it finds a block at least as large as the request, marks off the space being allocated, identifies the allocated block with the number of the requesting job and returns the address of the block to the caller. If the caller so specifies, MEMAL will zero all locations in the block before it is allocated.

When the space is no longer required, the address of the block is passed to the MEMAL subroutine TAKE. TAKE first checks that the address is indeed within the limits of MEMAL space and then determines whether the returned block is contiguous to any other block of available space. If so, the blocks are joined together and treated as one large block. If this were not done, the available space would soon be cut up into many small and relatively useless pieces. If a job crashes while running, system routine returns all the MEMAL space held by the job.

Timing. Two types of timing are required by an experimental control system. First, the experimenter may want to control the pace of the experiment, limiting a subject's response time, controlling the length of a stimulus presentation, etc. Secondly, he may wish to measure the subject's response latencies. The system under discussion contains two separate timing devices for these two purposes.

(1) The first system, time delay, has been mentioned previously. The experimenter may request an unconditional time delay, in which case the job will be suspended until the expiration of the time delay, or he may

place a time limit on the subject's response by requesting a response or a time delay. In this case, the job will be suspended until the occurrence of the response or the expiration of the time delay, whichever occurs first.

Delay timing is controlled by a clock with a cycle time of 2 milliseconds. For convenience, however, delays are requested in units of one-hundredth of a second. Since there are several jobs running in the system, there will usually be a number of time delays being counted down concurrently. All time-delay requests are placed in a queue ordered on the basis of the expiration time of each delay. The clock counts down on only the first delay in the queue. When a time delay request is received, it is compared with the length of the first time delay in the queue. If the new time delay is shorter, it is placed at the head of the queue. The time delay which was previously at the queue head is now second and has its length shortened to the difference between its original length and the length of the new time delay. For example, suppose that the first delay in the queue, delay A, had an original value of five seconds. One second after delay A was requested (its value is now four seconds), a one-and-one-half second delay, delay B, is requested. Delay B is placed at the head of the queue while delay A, with its value shortened to two-and-one-half seconds, is placed in the second position. One-and-one-half seconds later, delay B will expire and the clock will resume counting down on delay A. Delay A will then expire two-and-one-half seconds later or five seconds after its request was made. If a new time delay is longer than the first time delay, it is successively compared to each of the delays in the queue. Following each comparison which finds the new delay to be the longer, its value is decreased by the value of the delay with which it was compared. Eventually, the delay is placed in its appropriate position in or at the end of the queue with its value reduced to compensate for all the delays preceding it in the queue. If it is found that a new delay will expire at exactly the same time as an existing delay, the value of the new delay is increased by a single two millisecond clock unit. The queue itself consists of three cells in each job's JST, one cell which contains the adjusted delay value, one which contains the address of the immediately prior delay in the queue and one which contains the address of the next delay in the queue.

The clock simply decrements a value in a single register once every two milliseconds. This register contains the current value of the first delay in the queue. If a new delay is placed at the head of the queue, the value in the register is corrected accordingly. When the value in the register reaches zero, an interrupt is caused on API Channel 3. This causes control to pass to system routine Time Delay Over (TDOVER) which acts like a response device control routine to call SETUP to place the delayed job in the ready-to-run queue.

Since jobs may also request conditional time delays, a suspended job is frequently set up due to a subject response while its conditional time delay is still in the time delay queue. When SETUP determines that a job was suspended pending a time delay, whether this was the reason for the termination of its suspension or not, it calls a Time Delay Delink routine (TDDL) which removes the appropriate delay from the queue. If the removed delay was in the middle of the queue, the value of the next delay is increased by the value of the removed delay and it is linked to the delay which previously preceded the removed delay. If the removed delay was at the head of the queue, TDDL also starts the clock counting on the next delay in the queue.

(2) The second system used for timing of external events is considerably simpler. It consists of a thirty-six bit register which is incremented once every millisecond. This is treated as a "time-of-day" clock and is never altered other than constantly being incremented. The clock

is read into two words of memory specified by the caller whenever the appropriate IOT is issued. Since the value in the low order 18 bits is repeated only once every eight-and-one-half minutes, the use of the single low order word is usually sufficient for timing subject responses.

It will be recalled that a suspended job is placed in the ready-to-run queue whenever a requested subject response is made. If there are several other jobs in the queue, however, an appreciable length of time (on the order of a few hundredths of a second) might elapse between the time the response was made and the time at which the experimental job read the clock. For this reason, all of the stimulus and response device control routines provide options for reading the clock and storing the time-of-day at the exact time that a stimulus is presented or that a response is made. These stored times are then available to the experimental job whenever it runs again.

Operator Control and Error Detection

For the most part, the operation of the system is controlled by means of the control teletype and a package of system operator control keyboard routines (SOCK, Jackson, 1968). As was mentioned previously, the control teletype is assigned to API Channel Zero. In the event of a system crash, the control teletype is least likely of all the peripheral devices to be affected. Its various functions will be discussed in the order in which they might be used by the system operator.

Currently, the system itself is stored on disc and is loaded into the machine by means of a bootstrap paper tape. This bootstrap contains a short program which reads the system into core from disc and calls system routine ANFANG which initializes the system, starts Job Zero running and activates the control teletype. In the event that the system needs to be reinitialized (following system crash, for example), it can be done by means of the control teletype. The typed command AFNG calls ANFANG which destroys all existing JST's, initializes MEMAL space to include its full capacity, enables the appropriate A PI channels and starts Job Zero running.

For the most part, currently operating applications programs are stored on disc. System Routine DOODLE (Dynamically Operated On-Line Disc Loader, Jackson, 1969) enables the operator to read and write 8K fields to and from disc via the control teletype. The command DLST causes DOODLE to list the symbolic names of all disc files. Typing DRED File Name, Field Number causes the named file to be read into the the specified field. Likewise, the command DWRT File Name, Field Number results in the contents of the specified 8K field being copied onto a disc file which is then given the symbolic name specified. The command DDEL Symbolic Name deletes the named file from disc. Any given file may contain one or more experimental control programs, depending on the size of the individual programs.

Once a program has been read into core, one or more jobs must be created to use the program. This is done by means of a Create command as follows: CRET Job Number, Rank, Starting Address. This creates a JST for the job of the specified number, assigns the job the specified rank, records the address at which the job is to start running and leaves the job in the suspended state pending operator intervention.

Now the appropriate stimulus and response devices must be assigned to the job. Suppose that the program requires the use of a C R T and a keyboard. There may be two or three such pairs of devices in the laboratory and each device is identified by a Physical Unit Number. The program, in turn, refers to each device by a Logical Unit Number. Since two or more jobs may be using the same code-shared program, a unique Logical-Physical Unit Number match must be made for each job. This information is stored in a Conversion table by means of the control teletype Convert Add (CNAD) and Convert Delete (CNDL) commands. For example, the command CNAD Job Number, Unit Type, Logical Number, Physical Number constructs an entry in the Conversion Table for the specified job and the particular type of device. Subsequent calls on the device control routine from the experimental program will refer to this entry to convert the logical number supplied by the program to the specified Physical Unit Number. When a job is completed, the Conversion Table entry is deleted by a CNDL command.

When the experimenter is actually ready to begin the experiment, the operator will set up the suspended job via teletype command SETP as follows: SETP Job Number, Suspension Reason, Starting Address. As a result of this command, the job is placed in the ready-to-run queue as was discussed above. Alternatively, the Create and Set Up commands may be combined by means of the Job Go command: JBGO Job Number, Rank, Starting Address. This command creates a JST for the job and starts it running at the specified address. At some times, such as in the case of a program or experimenter error, it is desirable for the operator to be able to suspend a running job from the control teletype. This is done by means of the following command: SUSP Job Number. This will cause the job to be suspended at its current address pending operator intervention. At a later time, it can be restarted via the SETP command. A job may be completely terminated and removed from the system by means of the KILL command as follows: KILL Job Number.

As was discussed previously, system Job Zero runs when all other jobs in the system are suspended. Normally, a system Job Zero simply runs in a tight loop but at times it is convenient to transfer Job Zero's control from its normal line of code (called Twiddle) to the DEBUG routine which normally (when there is room) resides in the very top of core.

This routine allows the operator to examine or alter a running program. This transfer is effected from the control teletype by the command JOBZ DD3 where the 3 indicates that the DEBUG program is to be found at the top of Field 3. Job Zero can be transferred back to its normal code line by the command JOBZ TWD.

All system routines check parameters passed to them by experimental programs and other system routines for errors. When such an error is found, the routine which discovered the error calls the system error routine SYSERR. SYSERR determines whether the error concerns only the offending job or if a general system malfunction is indicated. In the former case, SYSERR simply suspends the offending job while in the latter, it halts the entire system. In either case, a coded description of the error is printed out on the control teletype.

Peripheral Equipment Control

The Central Executive System discussed above composes only about half of the total Executive System. The next largest component is the set of peripheral equipment control (PERP) routines. While it might be feasible for each experimental program to control the experimental stimulus and response devices directly, this would result in a large duplication of effort and would substantially increase the complexity of the individual experimental programs. Consequently, a package of control routines is made available to the experimenter to control all of the standard terminal devices.

All PERP routines are similar in that they are "re-entrant." That is, a single routine will control several devices of the same type. At any one time, it might be servicing two or more programs which in turn are each controlling two or more devices of that type. Since only one job can be running in the system at a given time, there cannot be two simultaneous requests on a PERP routine. Likewise, if all jobs in the system have the same rank, a PERP request will normally be completed before a second request is received. The major problem arises when jobs in the system have heterogeneous ranks. For example, a low ranked job may have made a request to a PERP routine just before the occurrence of an interrupt and subsequent set-up of a higher ranked job. The lower ranked job will then be suspended in the middle of the PERP routine. If the higher ranked job then makes a call on the same PERP routine, the routine will note that it is already in use. When this condition is detected, the routine calls system routine RENTRY (Buckwalter, 1966) which suspends the re-entrant job at the head of the PERP routine. The scheduler will then automatically return control to the lower ranked job which is in the middle of the PERP routine. If additional higher ranked jobs call the active PERP routine, they would also be suspended in the same fashion. When the PERP routine completes its work and is ready to return to the caller, it checks its re-entry queue. If there are any higher ranked jobs in the queue, the routine places a second call to RENTRY which then sets up all of the suspended jobs in the queue. The scheduler then automatically sorts out the running order of the jobs and returns control to the highest ranked job which has been waiting for the longest period.

A second point of similarity between most PERP routines is the GRAB feature. All subject terminal devices and some I/O devices must be grabbed by a job before the job can make use of that device. This prevents accidental interference of an on-going experiment by a new experiment which is just being set up. Once a particular job has grabbed a piece of equipment (by means of a special GRAB call on the device control routine which in turn calls system routine GRAB, Buckwalter, 1966), it retains exclusive control of that device until it has released it. If another job of equal rank attempts to grab the same device, the grab is denied and the Link is set when control is returned to the user to indicate that that device is not available. If a job attempts to use a device which it has not grabbed, it is suspended and a SYSERR message is printed out on the control teletype. It is possible for a higher ranked job to grab a subject terminal device (but not an I / O device) away from a lower ranked job. This feature was designed into the system to allow a human monitor to take over control of a student's terminal. It was envisioned that such a procedure might be useful in the developmental testing of C A I programs, but the feature has thus far never been used and it is likely that it will be dropped in future revisions of the system.

While all subject terminal devices must be grabbed, only some of the I / O devices incorporate the grab feature. The paper tape reader and punch and the printer must all be grabbed before they can be used. The magnetic tape and disc need not be grabbed. If a second job places a request to one of these devices while it is in use by another job, the second request is simply queued until the first request is completed as was discussed above under Multiple Entry.

Input/Output device routines. The system is supported by a number of I / O devices: paper tape reader and punch, card reader, magnetic disc, magnetic tape drive and printer. All of these devices except the paper tape reader are controlled by system routines which allow an applications program to read or write data.

PUNZIT (Buckwalter, 1969) controls the paper tape punch and handles all grab, release, and punch commands. Data to be punched are stored in a core buffer, the address of which is passed to PUNZIT. At the user's option, PUNZIT will return the buffer to MEMAL space following the completion of punching. The user has the option of passing PUNZIT the size of the buffer to be punched or setting an end-of-data flag following the last character to be punched. Tape may be punched in either an alpha-numeric or binary format. Finally, the user has the option of suspending until the completion of the punching operation or being allowed to continue with other work.

CARD (Buckwalter, 1967) processes all calls to the card reader: grab, release, and read. A separate call must be placed for each card to be read and the number of card columns to be read must be specified. Data read from the card is stored in a buffer location, specified by the user, in either binary, octal, or alpha-numeric format. Again, the user has the option of specifying whether or not he wishes to be suspended until the completion of the card read. The current routine was actually written for a previous card reader. A new routine will be implemented when the current card reader is interfaced through the multiplexer rather than via the AC as it is now.

DISC (Slaughter, 1968) is a system routine for reading and writing on the magnetic disc. Due to the short time which a job requires for a single disc transfer, there is no grab or release function. Successive calls are simply queued up by DISC and proceeded in the order of their occurrence. Higher ranked jobs are able to exercise a priority option which places their request at the head of the queue. A user may read or write in either word or byte mode on side zero of the disc. Only read commands are accepted for the protected, side one of the disc which is used for permanent storage of the system and currently operating programs. A minimum of one 32_{10} word sector may be read or written. The user has the options of having his buffer space returned to MEMAL on a write command and of suspending or not until the completion of either read or write commands.

MAGTAP (Buckwalter, 1967) allows a user to control the magnetic tape drive. MAGTAP is not actually an in-system routine. A true in-system routine will be written only after a battery of new drives is installed in late 1969. MAGTAP does not include the grab and release features nor does it provide the options of suspending the user or returning buffer space to MEMAL. The user is able to read or write on tape in odd or even parity, erase tape, and skip tape records or files in both the forward and backward directions.

PRINTR (Slaughter, 1968) is an in-system routine to control the line printer. The commands available allow a user to grab or release the printer, print a single line or move the paper to the head of the next page. Since PRINTR queues successive requests, a job may make repeated calls on PRINTR, each call corresponding to one line and then request suspension until the completion of printing the last line. The option of returning the print buffer space to MEMAL is also available.

Examples of a stimulus device control routine. While space does not allow a complete description of the software support of all of the different terminal devices, it may be informative to discuss one of these routines in some detail. The HYSPRJ (for high speed projector, Pethia, 1968) routine for controlling the Kodak RA - 950 Carousel Projectors is a typical example of the stimulus device routines. This routine handles the positioning of slides, the suspension of the caller (if desired), control of the projector shutter, and automatic position initialization of the carousel slide tray. HYSPRJ may be called by any job from memory fields 1 through 3.

When a job wishes to make a HYSPRJ call, the following sequence of instructions is required:

```
        LAC  X          /load the A C  with the appropriate
                         command

        EEM             /place the system in extend mode so that
                         a cross field jump may be made

        DPI             /disable the A P I
```

```
JMS * (HYSPRJ)     /jump to the head of the HYSPRJ routine in field 0
```

If the job is simply grabbing or releasing the projector, the contents of the AC (X in the example above) will contain a logical number of the desired projector and a zero or one, indicating grab or release respectively. If the job wishes to position a particular slide for showing, the AC would contain the command code 4 and a pointer to a parameter list. For the HYSPRJ routine, the parameter list consists of two words. Word one contains (a) a one-bit flag which indicates whether or not the slide is to be shown after it has been positioned, (b) the number of the projector, (c) a one-bit flag indicating whether or not the caller wishes to be suspended until the completion of the positioning, and (d) the number of the desired slide. Word two of the parameter list contains a pointer to the location at which the caller wishes to have the time-of-day stored when the slide is in position. If this word contains a zero rather than a pointer, no time-of-day is recorded.

If the caller asked that the slide be positioned but not displayed, he will make a later call requesting that the slide be shown. In this case, the AC will contain a command code of 5 and a pointer to the parameter list. Word one of the parameter list will contain only the number of the projector and word two will again contain a pointer to the location at which the time-of-day is to be stored.

Like most stimulus device and all response device control routines, HYSPRJ consists of two subroutines -- a request processing subroutines (RPS) and an interrupt processing subroutine (IPS). Flowcharts of these subroutines are shown in Figures 1 and 2. When a call is received by the HYSPRJ RPS, it first checks as to whether or not the routine is already in use. If so, the calling job is suspended and queued by a call on system routine RENTRY as discussed above. If not, then HYSPRJ checks the

HYSPRJ FLOWCHARTS

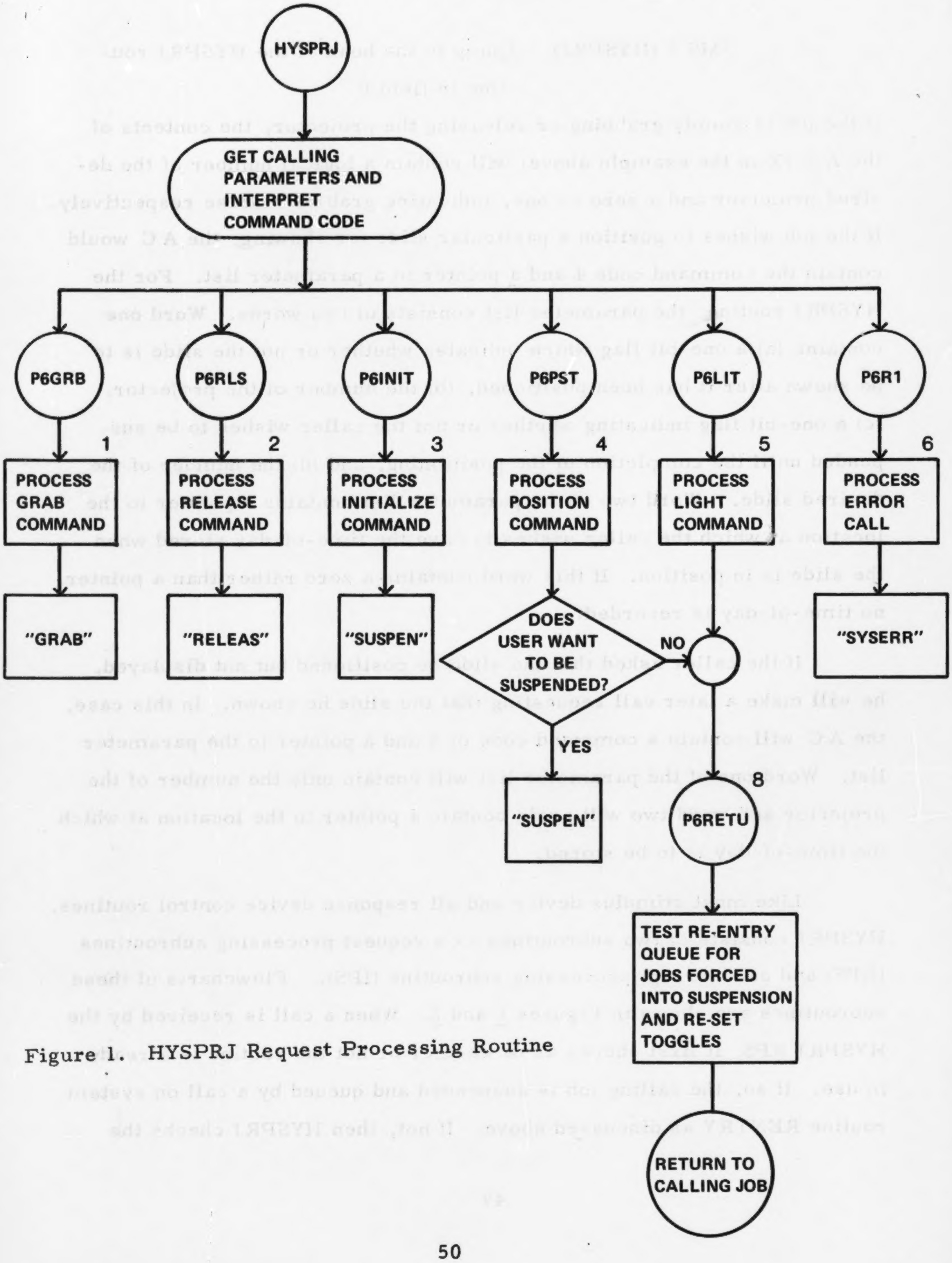

HYSPRJ
GET CALLING PARAMETERS AND INTERPRET COMMAND CODE
P6GRB
P6RLS
P6INIT
P6PST
P6LIT
P6R1
1
2
3
4
5
6
PROCESS GRAB COMMAND
PROCESS RELEASE COMMAND
PROCESS INITIALIZE COMMAND
PROCESS POSITION COMMAND
PROCESS LIGHT COMMAND
PROCESS ERROR CALL
"GRAB"
"RELEAS"
"SUSPEN"
DOES USER WANT TO BE SUSPENDED?
NO
YES
"SYSERR"
"SUSPEN"
8
P6RETU
TEST RE-ENTRY QUEUE FOR JOBS FORCED INTO SUSPENSION AND RE-SET TOGGLES
RETURN TO CALLING JOB

Figure 1. HYSPRJ Request Processing Routine

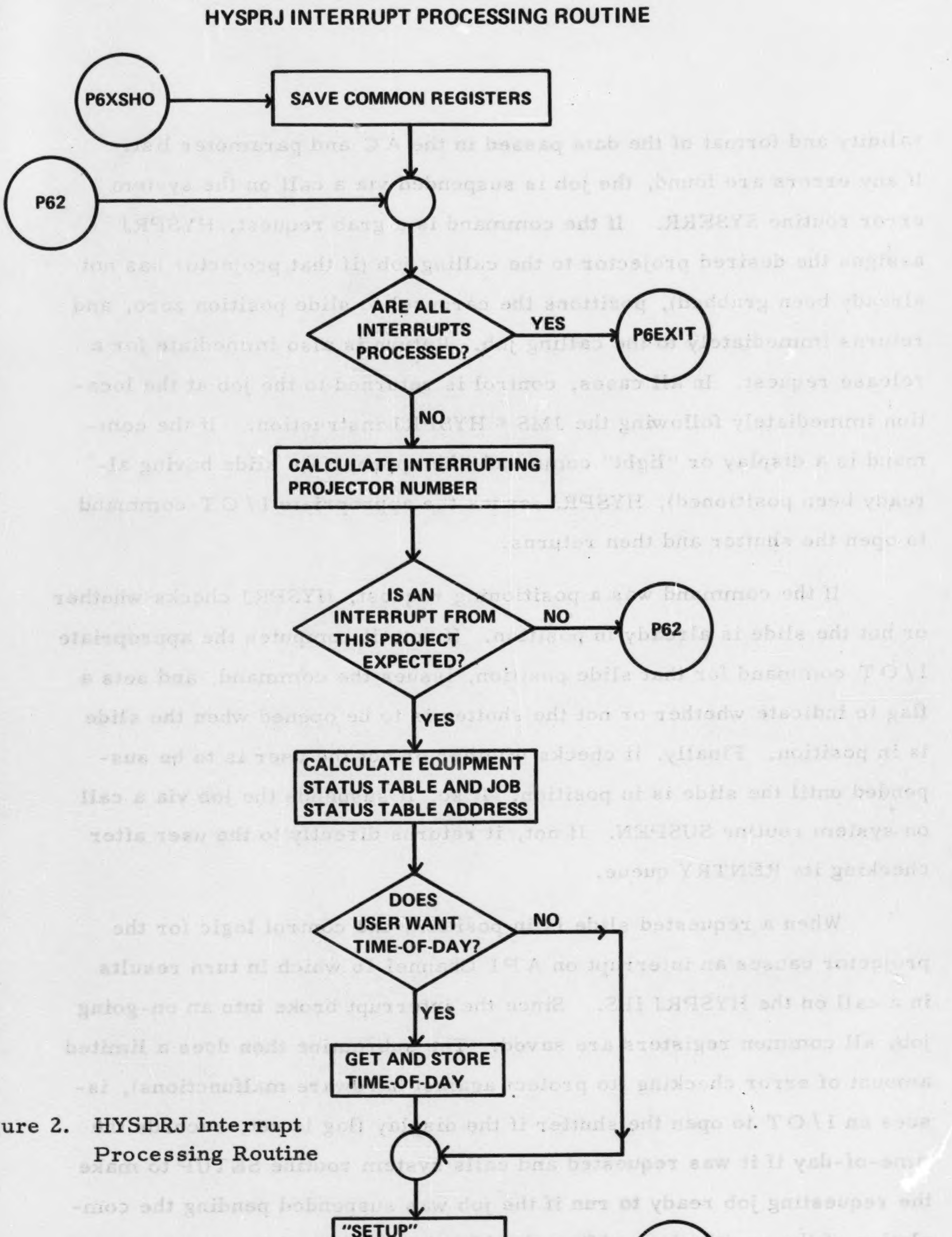

HYSPRJ INTERRUPT PROCESSING ROUTINE
P6XSHO
SAVE COMMON REGISTERS
P62
ARE ALL INTERRUPTS PROCESSED?
YES
P6EXIT
NO
CALCULATE INTERRUPTING PROJECTOR NUMBER
IS AN INTERRUPT FROM THIS PROJECT EXPECTED?
NO
P62
YES
CALCULATE EQUIPMENT STATUS TABLE AND JOB STATUS TABLE ADDRESS
DOES USER WANT TIME-OF-DAY?
NO
YES
GET AND STORE TIME-OF-DAY
"SETUP" SET UP USER FOR PRJ. SUSPENSION REASON
P62
RETURN TO INTERRUPTED PROGRAM
P6EXIT
RESTORE COMMON REGISTERS
DE-BREAK INTERRUPT CHANNEL
EXIT

Figure 2. HYSPRJ Interrupt Processing Routine

validity and format of the data passed in the A C and parameter list. If any errors are found, the job is suspended via a call on the system error routine SYSERR. If the command is a grab request, HYSPRJ assigns the desired projector to the calling job (if that projector has not already been grabbed), positions the carousel to slide position zero, and returns immediately to the calling job. Return is also immediate for a release request. In all cases, control is returned to the job at the location immediately following the JMS * HYSPRJ instruction. If the command is a display or "light" command (the appropriate slide having already been positioned), HYSPRJ issues the appropriate I / O T command to open the shutter and then returns.

If the command was a positioning request, HYSPRJ checks whether or not the slide is already in position. If not, it computes the appropriate I / O T command for that slide position, issues the command, and sets a flag to indicate whether or not the shutter is to be opened when the slide is in position. Finally, it checks whether or not the user is to be suspended until the slide is in position. If so, it suspends the job via a call on system routine SUSPEN. If not, it returns directly to the user after checking its RENTRY queue.

When a requested slide is in position, the control logic for the projector causes an interrupt on A P I Channel 16 which in turn results in a call on the HYSPRJ IPS. Since the interrupt broke into an on-going job, all common registers are saved. The subroutine then does a limited amount of error checking (to protect against hardware malfunctions), issues an I / O T to open the shutter if the display flag is set, records the time-of-day if it was requested and calls system routine SETUP to make the requesting job ready to run if the job was suspended pending the completion of the positioning. After checking for additional interrupts on Channel 16, the IPS restores the common registers, reactivates Channel 16, and returns to the interrupted job.

The above discussion is far from a complete description of the HYSPRJ routine. The parameter validity and format checking mentioned is rather extensive. Due to the number of options available to the user program, HYSPRJ is broken into a number of subroutines. The various combinations of the requested options result in a number of different possible paths through these subroutines. All in all, the routine occupies 1040_8 machine locations.

Other stimulus device control routines. Other PERP routines will be described only briefly. System routine SCREEN (Buckwalter, 1969) controls the various C R T displays. A user program can call SCREEN to (a) plot a single alpha-numeric character in storage mode, (b) display a text string in storage or dynamic mode, (c) plot an arbitrary list of points in storage or dynamic mode, and (d) erase a storage mode display. In storage mode, the display is plotted once and remains visible on the screen until it is erased. In dynamic mode, the points plotted fade immediately, and display visibility is maintained by constantly refreshing the display at a rate specified by the user. SCREEN's calling sequence is similar to the HYSPRJ calling sequence described above. Parameters are passed from the user program to SCREEN via the A C and a parameter list. Unlike HYSPRJ, there is no suspension option available to the user on display calls since relatively little time is required to plot a display (an average of 1 msec. per character). A user does have the option of suspending during an erase which requires 250 msec.

SCREEN consists of two major subroutines: (a) a request processing subroutine and (b) a system timing job. The RPS accepts all calls on SCREEN (namely grab, release, and display requests), and checks for multiple entry and parameter format and validity. For a grab, release, or storage mode display request, the RPS executes the request and returns to the caller. For an erase request, the subroutine sets the appropriate relay bit to start the erase, sets up the timing job and returns to or suspends the calling job. When the timing job's time delay expires, it terminates the erase and the RPS sets up the calling job if it was suspended. For a dynamic mode request, the RPS plots the display once and then sets up a timing job which autonomously continues refreshing the display.

CROW (Jackson, 1969) is the system routine which controls the random-access audio units. It has the usual calling sequence in which data is passed from the user to the system routine via the A C and a parameter list. A CROW request may be one of five commands: grab, release, pre-position the recording belt, play, or record. In addition, the caller has the option of requesting a suspension until the completion of his request. CROW consists of two main RPS and IPS portions. The RPS receives all calls from the user job, makes the usual checks, and then, in the case of a position, play or record request, starts the belt moving, and activates the record or play heads as necessary. It then either returns directly to the caller or suspends the calling job, depending on the caller's option. As was discussed earlier, holes at the edge of the belt, placed one second apart, are sensed photoelectrically and generate interrupts. The IPS checks each interrupt for validity and then determines whether or not the message is completed. If so, and if the caller requested the suspension option, he is set up at this time.

The relay buffers, used for controlling non-standard stimulus devices are relatively simple as compared to the devices discussed above and have a correspondingly simple control routine. The relay PERP routine consists of three subroutines, each serving a different function and each of which is called directly by the user job. Subroutine SETIMG sets the specified relays to the on position, CLRIMG sets the specified relays to the off position, and CNGIMG reverses the polarity of the specified relays. In calling each routine, the name of the particular relay buffer is placed in the MQ and the numbers of the appropriate relays in that buffer are passed to the subroutines in the AC. No parameter list is required, and there are no suspension options. Since there is no hardware representation of the states of the various relays, the relay buffer routine maintains a software record of the state of each relay.

Response device control routines. The first of the response device control routines to be discussed is KBUNLK (Cook, 1966) which controls the modified teletype keyboards which are usually used with the CRT displays. As is true of all of the response device PERP routines, KBUNLK consists of RPS and IPS portions of about equal complexity. Three commands are allowable: grab, release, and unlock (or activate) the keyboard. Parameters are passed to KBUNLK in the AC and a three-word parameter list. When the RPS receives a request, it stores the parameters being passed, and makes the usual checks. If the request is a grab or a release, return to the caller is immediate. If the command is an unlock request, KBUNLK determines whether the caller has exercised the option of requesting an "old" character. An old character is a subject response which was made at a time when the user job was not actually suspended waiting for a character, e.g., a response made after the expiration of a response time limit. If this option was requested, and if such a response was indeed made, the character code is complemented (to indicate an old character) and returned to the user job immediately. Otherwise, KBUNLK unlocks the keyboard and suspends the user job via system routine SUSPEN. It will be recalled that suspension is mandatory when a job is waiting for a subject response. If the user job requested a time limit on the response, KBUNLK requests that the job be suspended until the occurrence of a keyboard response or until the expiration of the time delay. In addition, it stores a zero in the calling job's JSTAC word. If the time delay runs out before a response is received, the job will be set up with a zero in the A C indicating that the time delay expired.

When a subject response is made, the IPS portion of KBUNLK first determines whether a job is actually suspended waiting for this response. If not, it simply stores the response as an old character. If there is a waiting job, the IPS records the time-of-day of the response in the location specified by the user job, stores the character code received in the job's JSTAC cell and calls system routine SETUP to place the job in the ready-to-run queue.

System routine TOUCHE (Pethia, 1968) controls the Touch Sensitive Surfaces. TOUCHE has the usual calling sequence in which data are passed from the user job to TOUCHE via the A C and a parameter list. The RPS portion of TOUCHE receives all calls from user jobs, makes the usual checks and determines whether the command is a grab, release, or activate request. Grab and release commands are processed and returned immediately. If the command is to activate a touch display, the time-of-day of the request is stored and that TOUCHE's active flag is set to "active". From the hardware standpoint, the Touch display is active at all times, so TOUCHE does not need to issue an I / O T to turn the device on.

When the subject makes a Touch response, it causes an interrupt on A P I Channel 2 which, in turn, calls the IPS portion of TOUCHE. The IPS checks that unit's active flag. If the flag is not set to active, the interrupt is ignored. If the unit is active, that is, if a job is suspended waiting for a response from that unit, the time-of-day of the response is stored and the coordinates are read and checked for legibility. If the coordinates cannot be interpreted, a code of 400000 is placed in the job's JSTAC word. If they can be interpreted, they are converted to a more convenient octal format and then placed in the job's JSTAC word. Finally, that unit's active flag is set to "inactive," and TOUCHE calls system routine SETUP to place the job in the ready-to-run queue.

The "in-house" teletypes, those used in the laboratory, are controlled by system routine BAGTEL (from bagatelle) (Jackson, 1968). BAGTEL's method of operation is essentially the same as that of the routines described above with the exception that it controls both stimulus and response devices. A user job may request (a) the printing of a single character or a text string, (b) the activation of the teletype to allow the typing of a single character, or (c) the activation of the teletype and the subsequent printing or "echoing" of the typed character.

System routine DPHONE (Pethia, 1969) handles all requests concerning dataphones and their associated teletypes. DPHONE has been constructed so that its calling sequence and, in general, its appearance to a user, is identical to system routine BAGTEL, described above. Therefore, any applications program written for the in-house teletypes can also be run on a remote teletype via the dataphone system. DPHONE differs from BAGTEL in that prior to issuing any dataphone I / O T, DPHONE checks whether the telephone line is clear to transmit. If not, the applications program is suspended for a short time delay and the line clear check is made again at the end of the time delay.

The in-house teletypes controlled by BAGTEL are all full-duplex. That is, a typed character is not automatically printed. A specific print I / O T must be issued for each character. Remote teletypes, serviced by DPHONE may be either full-duplex or half-duplex, in which case typed characters are automatically printed as they are typed. Like BAGTEL, DPHONE provides the option of echoing typed characters so that a full-duplex teletype may be treated as a half-duplex device.

Utility Programs

In addition to the system itself, there are a number of routines and program packages which are required for the preparation and maintenance of the system and applications programs and for the limited amount of data reduction which is feasible on the system. First, there are the utility programs which are used in conjunction with or in support of the time-sharing system. The most basic of these is CORFIL (Broadley, 1968) which is a basic bootstrap program that loads all four fields from disc. CORFIL stores a debugging program, described below, and the manufacturer-supplied FF and RIM paper tape loaders in each field. The T & D TRIO program, also described below, is stored in field one. The most common use of CORFIL is to prepare the machine to load a program that is currently being debugged and is stored on paper or magnetic tape. It is also used to load operational extra-system programs that are stored on magnetic tape.

SYSLOD (Pethia, 1969) is the system complement of CORFIL. It is also a bootstrap paper tape program that loads the system from disc. The basic system is stored in fields zero and one, and mini-execs are stored in fields two and three. SYSLOD then transfers control to system routine ANFANG which initializes the system, activates the control teletype, and starts Job Zero.

DEBUG (Fitzhugh, 1969) is an on-line, interactive debugging program operated from a teletype which has replaced the manufacturer-supplied DDT (Digital Equipment Corp., 1965). DEBUG allows an operator/programmer to examine, alter, and control systems and applications programs being tested and corrected. It can reside in the top of any of the four fields and can communicate with any of the other fields.

RECORD (Skillen, 1969) is an in-system utility which is used to control the random-access audio units for purposes of recording and editing audio messages. It is operated from a recording station consisting of microphone, headphones, modified teletype keyboard, and a CRT used to display information concerning message length, belt position, etc.

All systems and applications programs are currently assembled by means of the manufacturer-supplied MACRO-9 assembler (Digital Equipment Corp., 1967). MACRO-9 is a two-pass assembler written for the PDP-9 which produces binary object code from MACRO-9 assembly language. In addition, it provides a number of pre-defined macros (subroutines which may be incorporated into a program by simply referencing them) and allows the programmer to define his own macros. Both features are currently being used in systems and applications programs. Conditional assemblies are also provided in which predefined blocks of code may be selected for inclusion in a particular assembly at the option of the programmer. Finally, MACRO-9 is capable of generating relocatable code. That is, once a program has been assembled, it may be loaded and run at any core location. The macro and conditional assembly features are currently being used by both system and applications programs. The relocatable code feature is currently used only with extra-system programs run on the PDP-9. As it was supplied by the manufacturer, input to MACRO-9 was by means of punched paper tape, and output was in the form of paper tape and teletype listings. The assembler has been modified to accept punched card input, to use disc or magnetic tape as an intermediate storage medium, and to print program listings on the line printer.

The PDP-9 software supplied by DEC also includes an I/O Monitor System (Digital Equipment Corp., 1968a) and PDP-9 FORTRAN complier (Digital Equipment Corp., 1968b). The I/O Monitor, controlled from the PDP-9 teletype, facilitates the reading and production (listing or recording) of data in a number of formats. As in the case of the MACRO-9 assembler, it has been modified to use the disc, magnetic tape unit, card reader, and printer controlled by the PDP-7. The I/O Monitor and FORTRAN are used primarily for the listing and reduction of data generated by the CAI and experimental programs.

In addition, a number of out-of-system utility and diagnostic programs have been written to facilitate the out-of-system transfer of data between devices and to assist in the testing and maintenance of the I/O and subject terminal devices.

Higher-Level Languages

The development of a language suitable for CAI and/or behavioral experimentation has been one of the projects of LRDC. Initially, this work (Ramage, 1967; 1969) was of a basic, theoretical nature, and for the first two years, all applications programming was done in assembly language. It was assumed that eventually a language would be written which would be based on the results of the theoretical work being done. However, due to the departure of the key personnel involved, this work was slowed down. As a result, applications programming was being seriously retarded by the lack of a suitable higher-level language, and a language, even one that was only a temporary measure, was needed. The result of this decision was SKOOLBOL (Nemitz, 1968). SKOOLBOL (the name derives from COBOL, which it resembles in format) was developed for the purpose of immediate application to a specific system and for a specific purpose. There was no thought of writing a more generally applicable language suitable for other installations.

Since its development, SKOOLBOL has been continuously modified, but the initial concept has remained the same. It is intended to assist an applications programmer by reducing the complexity of the most common operations in a typical applications program. In most cases, this involves dropping some options in the device routine calls. Whenever relatively unusual conditions are required, the programmer lapses back into MACRO-9 assembly language. Typically, about twenty per cent of a SKOOLBOL program is written in assembly language. As long as the programmer follows the SKOOLBOL conventions, the structure of the language assures that his program is code-sharable, that is, two or more jobs can use the code at the same time. All equipment control calls have been simplified. For example, to print on a teletype, the programmer simply loads the address of the text buffer and calls SKOOLBOL routine TYPE. Textual materials to be displayed on the teletype or C R T may be formatted on punched cards and identified as text by a T in column one of the card. FORTRAN-like "DO Loops" are provided which allow a programmer to repeat subroutine calls a predetermined number of times. Boolian operators are available for conditional transfers. The programmer has the option of using the octal or decimal number system and may change from one system to another when it is convenient. SKOOLBOL also includes pre-programmed routines which are frequently required in applications programs such as a random-number generator and a routine to calculate means.

A SKOOLBOL language program (including blocks of assembly language) is first pre-assembled into MACRO-9 format code. It is then combined with the package of SKOOLBOL routines and assembled by the MACRO-9 assembler. The conditional assembly feature of MACRO-9 is utilized so that only the SKOOLBOL routines which are actually used by the program are assembled with the program. The final output is object code on punched paper tape and parallel listings of both the SKOOLBOL program and the resultant MACRO-9 assembly language program. The pre-assembly and assembly operations have been combined by temporarily storing the intermediate MACRO-9 code on disc so that, to all practical purposes, a SKOOLBOL program produces object code and the listing described above.

The major deficiency of SKOOLBOL for a behavioral research system is its lack of bit manipulation capability. It results, therefore, in inefficient table structures and the inefficient searching of tables. It has some very cumbersome aspects and does not include all of the operations that would be desirable. A major revision of the language is therefore scheduled for 1970 (Chadwick & Fitzhugh, in preparation).

Documentation

The importance of computer system documentation cannot be overemphasized. This is, perhaps, particularly to be emphasized for a system in a behavioral research environment. If a psychologist is accustomed to working with relay circuitry for experimental control, the problem may be even more severe. A few scraps of paper in a haphazard log book are usually sufficient to maintain and reconstruct all but the most complex relay rack circuitry. While it may appear obvious that more complete documentation is required for a computer system, the extent of the documentation which is actually required for efficient operation may not be at all obvious. In addition, when a system is in a university setting, the documentation procedures must take into account the mobility of the students and faculty who use and develop the system. It is all too easy to come to rely on the expertise of a particular person rather than insisting that he take time from his work to record what he knows.

Hardware documentation of the LRDC Computer Facility system is based on the manuals and drawings supplied by the manufacturers of each device. These are supplemented by detailed drawings by the Engineering staff of the interface between the device and the computer itself and of any modifications made to the device. The Engineering staff prepares a complete set of manuals and drawings for each device if the device was built or extensively modified in the LRDC shops. All of the above documentation is catalogued by a master drawing list. The operation of the hardware and its relation to software also composes one chapter of the LRDC Computer Facility Documentation Library.

Software documentation is a greater problem than hardware documentation. While it is possible to trace an undocumented circuit, it is usually easier for a Programmer to write a new program than for him to modify another Programmer's undocumented code. The LRDC Computer Facility Documentation Library consists primarily of software documentation, although as was mentioned above, one chapter is devoted to the hardware aspects of the various peripheral devices.

Whenever a new device is installed or a new program is completed, the Technician or Programmer responsible submits documentation to the staff Secretary. The Secretary edits, publishes, and distributes the material to all members of the Facility staff. An index of the current library contents is provided in Appendix A. This index is stored on punched cards and is updated, listed, and redistributed to all staff members on a monthly basis. Accompanying the index is a catalog of documentation abstracts consisting of from one to ten lines for each documentation note. These abstracts are also stored on punched cards and periodically updated and redistributed.

The form in which documentation is to be submitted has been standardized. Each of the major chapters--Hardware, Systems Software, Applications Software, etc.--has its own specialized documentation format, but the formats are all somewhat similar so as to facilitate the use of the documentation. An outline of one of these forms, for systems software documentation, is given in Appendix B. It is hoped that this outline will point up the more important aspects of such documentation.

Physical Plant

The LRDC computer room houses the PDP-7 and PDP-9 computers, line printer, card reader, magnetic tape drive, three console teletypes, random-access audio units, dataphone cabinet, a cabinet containing the subject terminal patch panels, two closed circuit TV monitors, a large magnetic tape filing cabinet (the top of which provides additional work space), and two files for manuals and paper tapes. The 450-square-foot room has a raised floor to provide a pathway for the cables connecting the various devices. To control the noise level in the room, particularly when the printer, card reader, or teletypes are running, a lowered ceiling of soundproofing material was installed and heavy draperies were hung on three walls; the fourth wall, facing a hallway, has glass windows for observation purposes. The room is cooled by a 36,000 BTU air conditioning unit mounted on the roof. This provides barely adequate cooling and a 50 to 60,000 BTU unit would be more satisfactory.

Entry to the computer room is provided by an access room which contains the magnetic disc, a desk for the operators, and cabinets and shelves for storage of supplies, parts, tools, and maintenance equipment. The various laboratories, which are all on the same floor, are connected to the computer room by cables carried in a cable trough mounted near the ceiling in the main hall which runs the length of the building.

The main laboratory (designated the CAI Classroom) is a 300-square-foot room which has been broken up into a number of cubicles by means of partitions of less than ceiling height. Three cubicles, each of 25 square feet, are designed to contain a compact subject terminal such as a CRT and keyboard. Two other cubicles each have a touch-sensitive surface mounted in the front wall of the cubicle and provisions for earphones and other stimulus and feedback devices which a particular experimenter might wish add. The remainder of the room contains the slide projectors used with the touch surfaces, a cabinet for storing slide trays and a teletype for use by experimenters to control operating programs. The room has a raised floor for cables connecting the terminals to the computer, and the ceiling, walls, and partitions are covered with soundproofing material. A heavy-duty window air conditioner has, so far, proven capable of dissipating the heat generated by the terminal devices. Subjects working in the Touch-Surface cubicles can be observed through one-way glass from an adjoining observation room. This room is currently being wired for audio so that observers can monitor the messages that the subjects are receiving. Most of the cubicles can also be viewed via closed-circuit TV monitored in the computer room.

Five other smaller rooms are wired for computer-controlled experimentation. None of these have raised floors, but each contains a junction box which terminates the cables from the computer and to which the particular terminal or terminals in that room are connected. For the most part, these rooms are used for experiments that are fairly short-lived and/or for which special purpose terminals are constructed. In each case, a TV camera can be mounted in the room if the experimenter so desires.

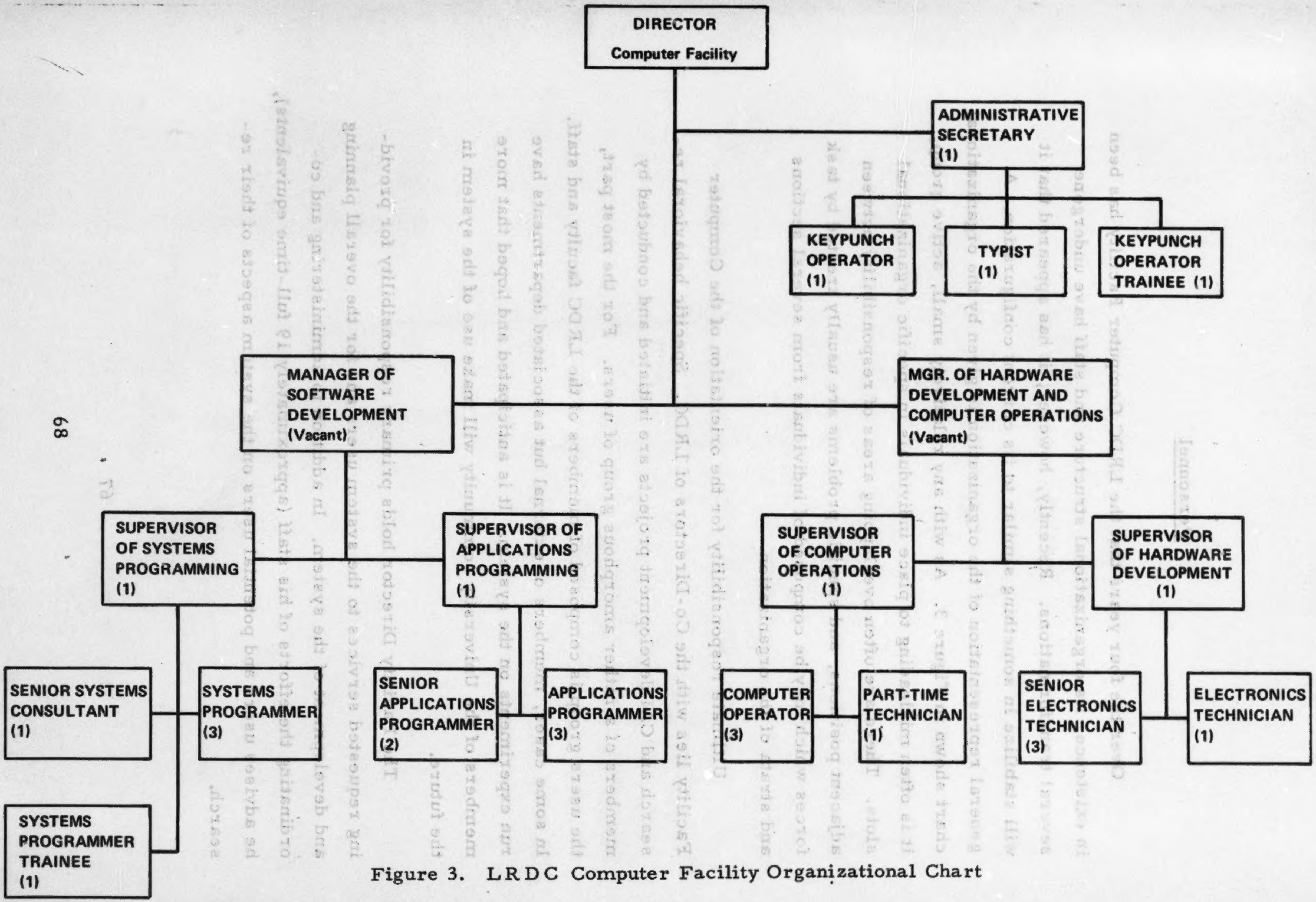
One small laboratory has been given a thorough soundproofing treatment and has been converted into a recording room. It is used by anyone who needs to make tape recordings but was designed specifically for recording and on-line editing of the random access audio unit belts. Finally, a 400-square-foot electronics laboratory is used for storage and for the design, construction, and maintenance of the subject terminals.

Personnel

Over the four years that the LRDC Computer Facility has been in existence, its organizational structure and staff have undergone several transformations. Recently, however, it has appeared that it will stabilize in something similar to its current configuration. A general representation of the organization is given by the organizational chart shown in Figure 3. As with any relatively small, active group, it is often misleading to place individuals in specific organizational slots. There are often overlapping areas of responsibility between adjacent positions, and specific problems are usually treated by task forces which may be composed of individuals from several sections and strata of the organization.

Ultimate responsibility for the orientation of the Computer Facility lies with the Co-Directors of LRDC. Specific behavioral research and CAI development projects are initiated and conducted by members of a rather amorphous group of users. For the most part, the users group is composed of members of the LRDC faculty and staff. In some cases, members of external but associated departments have run experiments on the system. It is anticipated and hoped that more members of the University community will make use of the system in the future.

The Facility Director holds primary responsibility for providing requested services to the system users and for the overall planning and development of the system. In addition to administering and coordinating the efforts of his staff (approximately 19 full-time equivalents), he advises users and potential users on the system aspects of their research.


LRDC COMPUTER FACILITY ORGANIZATIONAL CHART

DIRECTOR
Computer Facility
ADMINISTRATIVE SECRETARY (1)
KEYPUNCH OPERATOR (1)
TYPIST (1)
KEYPUNCH OPERATOR TRAINEE (1)
MANAGER OF SOFTWARE DEVELOPMENT (Vacant)
MGR. OF HARDWARE DEVELOPMENT AND COMPUTER OPERATIONS (Vacant)
SUPERVISOR OF SYSTEMS PROGRAMMING (1)
SUPERVISOR OF APPLICATIONS PROGRAMMING (1)
SUPERVISOR OF COMPUTER OPERATIONS (1)
SUPERVISOR OF HARDWARE DEVELOPMENT (1)
SENIOR SYSTEMS CONSULTANT (1)
SYSTEMS PROGRAMMER (3)
SYSTEMS PROGRAMMER TRAINEE (1)
SENIOR APPLICATIONS PROGRAMMER (2)
APPLICATIONS PROGRAMMER (3)
COMPUTER OPERATOR (3)
PART-TIME TECHNICIAN (1)
SENIOR ELECTRONICS TECHNICIAN (3)
ELECTRONICS TECHNICIAN (1)

Figure 3. LRDC Computer Facility Organizational Chart

The Manager of Engineering is a graduate electrical engineer. He is responsible for the day-to-day operations of the Facility as well as the design, implementation, and maintenance of the hardware system.

The Operations staff maintains control of the system during normal daily service. This consists of running preventive maintenance and system tests beginning at 7:00 a.m., preparing the system and peripheral devices for subject runs, monitoring the system during such runs, assuring that subject-generated data files are stored appropriately, and running assemblies and data reduction programs during a daily operations period. In addition, it is desirable that the operators be able to quickly diagnose system malfunctions to a sufficient degree to be able to notify the appropriate member of the Engineering or Software staffs. The Supervisor of Operations, a Senior Electronics Technician, is present during the early morning system checks and during the peak service hours. He is supported by two less highly trained operators who continue operator coverage until the completion of the daily operations period, usually about 7:00 p.m. The evening hours are devoted to program debugging, and usually only the Programmer involved is present.

The Hardware staff covers several areas of responsibility. They maintain the two computers as well as all of the attached I/O and subject terminal devices. In some cases, they design as well as construct or modify new terminal devices, construct and maintain the interfaces between the PDP-7 computer and the peripheral devices, and procure the supplies and equipment required for their work. While each of the three Senior Electronics Technicians in this group has one or more areas of specialization, they cannot be neatly categorized as to main-frame versus peripheral devices, or design and construction versus maintenance. Consequently their various functions have been grouped together in the organizational chart. They are assisted by two Junior Electronics Technicians.

The position of Software Development Manager is currently open. The position was previously held by the current Facility Director, and he continues to manage the programming effort. The Software Manager is responsible for the preparation and maintenance of all of the systems and applications used by the Facility. In addition, he coordinates his efforts with the Engineering Manager for the further development of the system.

The Supervisor of Systems Programming directs and coordinates the growth, revision, and maintenance of all systems software and the majority of the utility programs. While operating under the general direction of the Software Manager, he is responsible for the detailed functions of the time-sharing system and peripheral equipment control routines. His staff currently consists of a part-time Senior Systems Consultant who has been involved with the system from its inception and one full-time Systems Programmer. Due to an anticipated increase in systems work, two additional Systems Programmers will be added in the near future.

Due to the complexity of applications programming (which is due, in turn, to the flexibility of the system, the variety of terminal devices, and the lack of a satisfactory higher level language) very few of the system users writer their own experimental or CAI programs. The Applications Programming staff consists of two Senior Programmers and two programmer trainees under the direction of a Supervisor of Applications Programming. The Applications Supervisor has primary responsibility for the design and implementation of the programs requested by the various experimenters. In most cases, the Supervisor designs the major aspects of the program on the basis of his interaction with the experimenter. The actual coding, debugging, and documentation of the program is then done by a member of the Applications staff under his supervision.

The Administrative Secretary, shown as reporting to the Facility Director, actually serves the entire Facility staff. In addition to the usual matters of correspondence, appointments, etc., she publishes and maintains the Documentation Library, does the detailed scheduling of system use, generates weekly reports to the users concerning system operations, and supervises a full-time keypunch operator.

In addition to its own staff, the Facility also draws support from several other groups within LRDC. The Center maintains general purpose shops capable of constructing electrical and electronic devices as well as carpentry, metal, and plastic work. In many cases, these shops supply special purpose terminal devices for particular experiments as well as constructing some of the subject terminals which contain the devices. A substantial portion of the work of the Center photographic laboratory is concerned with the preparation of slides used in the on-line projectors. Due to the accuracy of registration required when slides are projected onto a touch-sensitive surface, the photographers have had to develop fairly sophisticated techniques for photographing stimulus materials and mounting the slides.

Experimental Control Programs

Descriptions of 12 of the programs used to control experiments run during the last two years are provided on the following pages. A brief description of each experiment is given as well as descriptions of the control programs. The program descriptions are far from detailed, but an attempt has been made to demonstrate a variety of computer control applications and to emphasize the more interesting aspects of each of the different programs.

While most of the experiments might be described as basic learning studies, some are more nearly experimental CAI programs. Two major experimental CAI programs, a spelling program, and a numeral discrimination program, have not been described. Both of these programs are quite extensive and any attempt to describe them in the space available would be extremely cursory. Other than the fact that they required very large data bases for their operation and that their program logic was more extensive and complex, these programs illustrate few control aspects that are not also illustrated by the shorter programs.

Program: Angle Discrimination

Experimenters: Alex Siegel, Robert Glaser, Jacqui Held

Programmer: Raymond McKnight

Purpose: It has been shown that young children (ages three to four) have great difficulty in learning to discriminate between oblique lines oriented in opposite directions. In addition, earlier work in this laboratory had demonstrated that discrimination learning was retarded if incorrect responses were immediately followed by the presentation of the next learning trial. It appeared that the incorrect responses were inadvertently reinforced by the stimulus change produced by the presentation of the next item. This study attempted to demonstrate that such a difficult discrimination could be learned if the response history was appropriately controlled. Two methods of stimulus presentation (fading, which minimized error responses, and the classical contrast method, which allowed frequent errors) and two response contingencies (delay, in which an incorrect response did not result in a stimulus change, and no delay, in which the reinforcing effects of stimulus change were possible) were evaluated.

Subjects: Children four years of age

Program Characteristics: Several versions of this program were run, all differing in some detail. The more general aspects of the control program will be discussed. A series of 50 slides was presented by a single slide projector. The child responded by pressing a round translucent window on which the stimulus was displayed. During different phases of the experiment, both one and two windows (successive and simultaneous discriminations) were used. Pressing on the window triggered a micro-switch which was treated by the system as a single element of the touch-sensitive surface. Reinforcement was provided

by a marble dispenser. The experimenter started the program by means of a teletype in the room with the child and then observed from a one-way vision window. Data were recorded on punched paper tape.

The ANGLE program was quite simple and only slightly response-contingent. Program flow under one of the experimental conditions, a simultaneous discrimination in which the subject was required to make a correct response, was the following: A slide was shown and the subject was given ten seconds in which to make his first response. If no response was made or if the response was incorrect, an incorrect-response counter was incremented by one, and a second ten-second period was begun. When a correct response was made, the subject was reinforced with a marble, the slide was turned off, and a four-second inter-item interval was begun. The latency of all responses, correct and incorrect, was recorded.

Reference: Siegel, A., & Glaser, R. Mirror-image discrimination learning in young children. Report in preparation.

Program: Sum and Recall

Experimenter: Charles A. Perfetti

Programmer: Martin Chadwick

Purpose: This experiment composed one portion of a study investigating immediate recall of sentences as a function of syntactic depth and lexical density. A sentence of either high or low depth and high or low density was presented to the subject. This was followed by an intervening task of summing a series of numbers. The subject was then asked to reproduce the sentence to the best of his ability. An earlier portion of the study presented the sentences and numbers aurally, using a tape recorder, and the subject wrote down his answers. The Sum and Recall program presented the sentences and numbers visually.

Subjects: College students

Program Characteristics: Materials were presented on the CRT, and subjects responded by typing their answers on a modified teletype keyboard. The program was completely linear and did not branch as a function of the subject's responses. In addition to variation in sentence depth and density, two other experimental conditions were involved. Under the "whole sentence" condition, a ten-word sentence was displayed on the CRT for ten seconds. This was followed by presenting a series of seven randomly generated numbers (ranging from 1 to 12) one at a time over a period of ten seconds. The message "SUM AND RECALL" was then displayed. The subject first typed in his sum, and indicated the end of his response by typing an asterisk. He then typed in the sentence as he recalled it, again terminating his response with an asterisk. The subject's responses were displayed on the CRT as he typed. Typing the asterisk caused the CRT to be erased. No feedback was provided and the program immediately proceeded to the next item.

Under the "partial sentence" condition, the ten words in the sentence were presented one at a time at a rate of one per second. After a word was presented, it remained on the screen so that, at the end of ten seconds, the complete sentence was displayed for a period of one second. This procedure was employed as an attempt to create a condition lying between the serial presentation of the earlier aurally presented sentences and the simultaneous presentation of the "whole sentence" condition. In other respects, the "whole sentence" and "partial sentence" conditions were identical.

Data were punched out during the subject run and consisted of the correct sum of the numbers, the subject's summing response, the stimulus sentence, and the subject's reproduction of the sentence.

Reference: Experimental results are currently being evaluated.

Program: PALL I (Paired-Associate Learning Latency I)

Experimenters: Wilson A. Judd, Robert Glaser

Programmer: Wilson A. Judd

Purpose: Response latency was investigated in a paired-associate task as a function of training procedure (comparison of the anticipation and study-test paradigms) and information transmission requirements (an eight-item stimulus list mapped onto two, four, or eight response alternatives) during both acquisition and overlearning. The data were treated on an item-by-item basis and analyzed relative to the trial-of-last-error (TLE) for each item.

Subjects: College students

Program Characteristics: Stimuli were CVC's presented one at a time on a CRT display. Subjects responded by pressing pushbutton keys mounted in a semicircle on a specially constructed panel. The matching of the keys to the stimuli was indicated by illuminating a pilot lamp next to the correct key while the CVC was displayed on the CRT. The stimulus materials and a set of 16 different stimulus-response matchings were all pre-stored in the program. The different experimental treatments and specific stimulus-response assignments were selected by entering a coded subject number via teletype. Item presentation order within trials was determined by a random number generator. Response latencies were measured and recorded under all experimental conditions.

Separate response records were maintained for each item. When an item reached a criterion of six successive errorless trials, it was tagged as learned and any subsequent errors were ignored. The experiment terminated ten trials after the last item in the list reached criterion. This procedure assured that all items received at least 16 trials of practice after the TLE, that is, the trial preceding the series of six successive

errorless trials. All experimental conditions included a warm-up list during which the subject was allowed only three seconds in which to make his response. Response time was unlimited during the experimental lists.

Under the anticipation paradigm, the program flow was as follows: The onset of a .5 second auditory warning signal occurred 1.5 seconds prior to the beginning of a trial, where a trial consisted of one presentation of the complete list. A stimulus item was displayed on the CRT. When a subject responded, or after three seconds had elapsed in the warm-up list, the pilot lamp next to the correct key was illuminated. Two seconds later the lamp was turned off and the display was erased. Following a 1.5 second inter-item interval, the next stimulus was presented, etc. Successive list presentations were separated by a four-second inter-item interval. The study-test procedure was made as similar as possible (in terms of timing, etc.) to the anticipation procedure.

Data were punched out on paper tape during the experiment. A data reduction program which rearranged the data into a TLE relative matrix and listed the matrix and trial means on a teletype was run as a background job. Thus, data produced by one subject could be reduced while the next subject was being run.

Reference: Judd, W. A., & Glaser, R. Response latency as a function of training methods, information level, acquisition and overlearning. Journal of Educational Psychology Monograph Supplement, 1969, 60(No. 4, Part 2).

Program: PALL II, III

Experimenters: Wilson A. Judd, Robert Glaser

Programmer: Ronald Confer

Purpose: Variability in response latencies had been found to be a severe problem in the previous PALL equipment. These two studies investigated potential methods of reducing this variability. This was done by (a) manipulating the response time limits in a warm-up list, (b) placing the inter-item interval under subject control, and (c) measuring the latency of response onset as well as response completion. Only the study-test paradigm was used. Materials consisted of eight CVC stimuli and eight response keys. The stimulus and response devices were the same as those used in PALL I, except for the addition of a "home position" key in the center of the response key array.

Subjects: College students

Program Characteristics: Since these were basically exploratory studies, the program was designed to permit the experimenter to vary many of the experimental conditions by means of values typed in from a teletype. While this was quite useful, it proved to be fairly confusing and laborious. For the most part, however, the program was similar to the PALL I program in that individual item records were maintained and items were trained to a TLE criterion. The PALL II program differed from PALL I primarily in the way it was paced.

During the study phase of each trial, the stimulus and the correct pilot lamp were displayed until the subject pressed the home key or until the expiration of a maximum display time determined by the experimenter. During the test phases, the stimulus was displayed when the subject pressed the home key. This allowed the subject to determine the time of stimulus presentation and also assured that his finger was in the home position at the time of stimulus onset. Subjects were instructed to hold the home key down until they were ready to make a response and to then make their response quickly. During the warm-up list, limits were placed on the time which the subject was allowed to hold down the home key and the time from the release of the home key to the depression of one of the response keys. If either time limit was exceeded, the item was counted wrong and the program went on to the next item. Latencies were measured from the time of stimulus presentation to the release of the home key (response onset) and to the depression of a response key (response completion). There were no time limits in effect during the experimental list but subjects were not informed of this. If keys were pressed or released in other than the required order indicating that the subject was using more than one finger to respond, the display was erased and the item was presented again.

Reference: Judd, W. A., & Glaser, R. Variability of response latency in paired-associate learning as a function of training procedure. Technical Report 9. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1970.

Program: PALL IV

Experimenters: Wilson A. Judd, Robert Glaser

Programmer: Wilson A. Judd

Purpose: The previous work, described above, had indicated that response latencies during overlearning might be indicative of how well individual items are retained over a period of time. Pilot work was conducted to determine the conditions under which a desirable, intermediate degree of retention was obtained, and a final experiment was run which attempted to demonstrate a relationship between overlearning response latency and subsequent retention.

Subjects: College students

Program Characteristics: The training procedure used was essentially the same as that described for PALL II, except that no time limits were placed on any of the subject's responses. Separate response onset and completion latencies were measured but only the full S-R latencies were used. The self-pacing procedure described for PALL II was retained. The same program was used to control the initial training session and the subsequent retention testing and relearning session.

The major point of interest about this program is its flexibility and the method used to define the experimental conditions. As was mentioned, a series of pilot studies was run which required a variety of experimental conditions. In order to facilitate the variation of the experimental conditions, an attempt was made to write PALL IV as a more general purpose paired-associate learning program. All input to the program was on punched cards, read at run time. First, an ID card provided the subject number, session number (for conditions under which each subject was run a number of times), and the number of paired-associate lists to be learned during that session.

Each list was headed by a parameter card giving the conditions under which that list was to be presented. The conditions available were as follows: (1) How long was practice to be continued? The experimenter had the option of running for a set period of time (specified in seconds), for a specified number of trials, or to an item criterion of a specified number of errorless responses. (2) If the learning criterion option was selected, the experimenter could specify the number of additional overlearning trials (if any) which items were to receive once they had reached the learning criterion. (3) Once an item reached the specified criterion, it could be dropped or retained in the list. Practice could continue until a specified number of items had been dropped or until all items in the list had reached criterion. (4) What time limits, if any, were to be placed on the onset and completion aspects of the response as described under PALL II. (5) The experimenter had the option of skipping the study phase of the first trial and starting with a retention test. (6) Finally, the number of items in the list was specified.

Following the parameter card was a deck of cards specifying the list, one card for each item. Columns one through eight of each card contained the stimulus, any word of up to eight letters, and the corresponding response number was punched in columns nine and ten. This scheme proved to be quite satisfactory. On several occasions, pilot work evaluated in the morning suggested experimental changes which were instituted with subjects run that afternoon. There is no doubt but that the procedure could be improved, but the scheme does appear to be a step in the right direction.

Data were recorded on disc during the experiment and punched out on paper tape by a background job following the completion of that subject's run.

Reference: Judd, W. A., & Glaser, R. Response latency as a correlate of retention. Report in preparation.

Program: TAP II

Experimenter: Charles A. Perfetti

Programmer: Raymond McKnight

Purpose: TAP II investigated the ability of children of various ages to deal effectively with grammatical transformations. The subject was presented with a stimulus sentence such as "The boy hit the ball" and was then asked to select the correct sentence from two alternatives such as "The ball was hit by the boy" and "The ball hit the boy." The subjects were assisted in their discriminations by three different degrees of visual prompting, conditions 1, 2, and 3 of the experiment.

Subjects: Children four to eight years of age

Program Characteristics: Auditory stimuli were presented by the random-access audio units. The subject responded by touching a section of a slide displayed on the Touch-Sensitive Surface. The experimenter controlled the program by commands typed on a teletype adjacent to the Touch Surface.

Under condition 1, the subject was instructed to touch the correct scene and then heard the stimulus sentence while a slide was displayed which showed both the action of the stimulus sentence and its reversal. Any responses which were not in the area of one of the two scenes were ignored. If no "legal" response was made witin the allotted time, the instruction was repeated. If the child did not respond within the second response period, the program proceeded to the next frame. If the subject made an incorrect response, he was given an audio message that his response was wrong and the instruction was repeated. The program then moved to the next frame regardless of the correctness of his second response. A correct response was reinforced by a bell and the next frame was presented immediately.

Experimental conditions 2 and 3 were particularly interesting since they gave the subject several different opportunities to make a response. Each frame in condition 2 consisted of an audio message and four slides. A sentence was presented in the presence of slide 1 which showed a scene depicting the action in the sentence. At the completion of the sentence, slide 2 was displayed which showed a picture of a cat and a scene depicting either the action of the stimulus sentence or its reversal. A second audio message was then played which consisted of "The cat says --" and either a correct transformation of the sentence (if the scene displayed depicted the action of the stimulus sentence) or the reversal of the stimulus sentence. The child was then given an opportunity to touch the display if he thought that it was the equivalent of the stimulus sentence. If he made a response and was correct, the program proceeded to the next frame. If he made an incorrect response, he was told that he was wrong, the sentence was repeated and the program proceeded to the next slide. If he did not respond within the allotted time, the program simply went on to the next slide. Slide 3 showed a picture of an elephant and a scene depicting either the action of the stimulus sentence or its reversal. If slide 2 was correct, slide 3 was always incorrect, and vice versa. The procedure followed for slide 3 was the same as that for slide 2 except that if the subject made an incorrect response, the program recycled through slide 2 and its accompanying audio message and then returned to slide 3.

If the subject responded to neither slide 2 or 3, he was shown slide 4 which included pictures of a cat, an elephant, and the action depicted in the stimulus sentence. He was then instructed to touch the animal which was correct. A correct response resulted in reinforcement and presentation of the next frame. An incorrect response caused the program to recycle through slide 2 of the current frame.

Condition 3 was the same as condition 2 except that the scene depicting the stimulus sentence or its reversal was not included in any of the slides.

Reference: Experiment still in progress.

Program: **Letter Discrimination**

Experimenters: **J. Michael O'Malley, Robert Glaser, Lauren Resnick**

Programmer: **Martin Chadwick**

Purpose: This study investigated the effects of response dimension pre-training and list length on the acquisition of a multiple discrimination by pre-school age children. The multiple discrimination problem is represented by such tasks as naming of letters, numbers, colors, etc. The particular discrimination task used in this study was the recognition of an auditorially presented letter among four visual response alternatives. Pre-training was on either the relevant dimension, shape, or an irrelevant dimension, color. List length was varied by training the subject to discriminate eight letters in subsets of two, four, or eight. The dependent variables were pre- and post-test score and the number of errors during acquisition.

Subjects: Pre-school age children

Program Characteristics: The program was presented on a Touch-Sensitive Surface using two slide projectors, a random-access audio unit and a teletype for use by the experimenter. Data concerning the subject and the experimental conditions of that session were entered by the teletype at the start of each session.

There were three phases to each session except the last. The first session began by training the subject to use the Touch-Sensitive Surface. This was followed by a pre-training phase in which the subject was exposed to the letters he would learn in that session and was trained to attend to either the color or the shape of the letter. This was followed by the training phase proper in which the name of the letter was finally introduced and the color cues to the letters were dropped. In sessions two and three, the touch-training phase was replaced by a post-test on the letters learned in

the preceding session. Session four consisted of only the post-test. The touch-training, pre-training, and training phases were all divided into two sub-phases, identification and recognition. Since the procedures used in the sub-phases were roughly equivalent in the different phases, only the touch-training phase will be described.

Each of the eight frames in the identification sub-phase began by displaying a slide showing only a colored square near the top of the screen. When the subject touched this square, it was immediately replaced by a slide, projected by the second projector, which displayed a collection of animals. An audio message such as "Touch the cow" was then presented. A correct response resulted in the re-presentation of the square which, in turn, led to the next slide. If the subject responded incorrectly, a red circle, projected from the other projector, was displayed around the correct animal. This was accompanied by an audio message such as "Touch the cow; the animal with a red circle around it." If the subject's second response was incorrect, this message was repeated. A third error resulted in a call to the experimenter's teletype asking for a decision as to whether to proceed or to terminate the subject.

The recognition sub-phase reviewed the same eight animal identifications using roughly the same logic. In this case, however, the sequence of frames was repeated until all of the frames had reached a criterion. Criterion for a frame consisted of n successive correct first responses to that frame where n was specified by the experimenter at the beginning of the session. As each frame reached criterion, it was dropped from the series.

Reference: O'Malley, J. M., Glaser, R., & Resnick, L. Response discrimination pretraining and list length in learning a multiple discrimination. Report in preparation.

Program: Preferences

Experimenters: Sigmund Tobias, Robert Glaser

Programmers: Martin Chadwick, William Schmeidlin

Purpose: This study investigated the question of whether, once an individual has been exposed to two instructional methods, his preference for one method leads to higher achievement with that method than with an alternative method. Subjects were taught to spell 28 words with one of two methods: (1) a visual method in which the subject heard the word and then saw it printed out on a teletype, one letter at a time, and (2) an aural method in which the subject heard the word pronounced and then spelled out aurally, one letter at a time. Under either treatment, the subject was then required to type the word.

The experiment used four experimental groups. Group 1, the preference group, selected their preferred instructional method at the start of each of four training periods. Subjects in the second group were used as yoked controls for the preference group, each subject in the yoked group receiving the same instructional treatment as a subject in the preference group. In addition, an attempt was made to equate the spelling words of two matched subjects as closely as possible. Group 3 received only the visual training method and Group 4 only the aural method.

Subjects: Fourth-grade school children

Program Characteristics: The experiment was run at the Oakleaf Elementary School, an elementary school associated with LRDC and located in a Pittsburgh suburb. The subject terminals consisted of a teletype, controlled by dataphone from the Computer Facility, and audio transmitted over telephone lines from the random-access audio units.

Spelling words were drawn from a pool of 70 sets of three words each, words within a set being selected so as to be very similar to each other. Each of four sessions began with a pre-test for all subjects. The subject heard an audio message such as "The cow is in the field. Spell the word cow." If the subject misspelled the word, it was used as one of the experimental words. If he spelled the word correctly, he was tested on another word from the same set. If he was able to spell all words in a set correctly, the program moved on to another set. This procedure continued until one misspelled word was found from each of seven sets.

All subjects were next given a demonstration of both the visual and aural training methods. The preference group subjects were then given their choice of training method, indicating their preference by answering a question printed out by the teletype.

The training procedure followed the same format for all groups. The program cycled through the seven words twice. The word was presented by the visual or aural method and the subject was instructed to type it. If the subject misspelled a word, it was presented again by the same training method. He was allowed up to ten attempts before the program moved on to the next word. The session ended with a post-test using the same logic as the pre-test except that the subject was tested on only the seven words on which he had been trained.

One of the more interesting aspects of this experiment was the method by which spelling words were selected for the yoked group. Data for all subject runs were stored on magnetic tape. Prior to any session, data were transferred from tape to a data file on the magnetic disc. A preference subject was run through all four sessions before his yoked subject began the program. During the pre-test for a yoked subject, he was first given a word which his matchmate had misspelled. If the yoked subject also misspelled that word, it was used as one of his spelling words. If the yoked subject spelled the word correctly, the program tried one of the two other words in the same set. If the subject could spell all of the words in the set, the program began searching those sets to which the subject's preference group matchmate had not been exposed. In doing so, the program considered only those words which were within one letter of being the same length as the word which it was attempting to match. When a word was found which the subject could not spell, the program moved on to the next word misspelled by the subject's matchmate. When seven such words had been determined, the training procedure was begun.

The data file stored on disc was constantly updated as the subjects responded. At the end of the session, the updated file was copied onto magnetic tape and a summary of that session's results was punched out on paper tape.

Reference: Tobias, S., & Glaser, R. Effect of pupil choice of instructional method on achievement and attitude. Report in preparation.

Program: Digit Memory Span

Experimenter: James G. Holland

Programmer: Marjorie Jackson

Purpose: This was one experimental CAI program employed in a project on the training pre-school skills. Series of digits were presented auditorially to which the subject responded by touching the appropriate digits, in the right order, on the Touch-Sensitive Surface. As the series proceeded, the series increased in length and complexity. Early items at each level of difficulty used stress and pauses to help the subjects segment the series. Complexity was increased by changing the order of the digits first at the beginning of the series, then at the end, and finally in the middle of the series.

Subjects: Pre-school age children

Program Characteristics: The number sequences were presented via the random-access audio units. Following the completion of the audio message, the numerals 1 to 9 were displayed on one row of the Touch-Sensitive Surface. The subject responded by touching the numerals in the appropriate order. It was found that performance was facilitated by an immediate auditory signal whenever the subject touched the screen and his response was registered. If he completed the sequence correctly, the program moved on to the next frame in the series. If he made an error at any point in the series, the display was immediately turned off and the audio message was repeated. The subject continued working on the slide until he made a correct series of responses, at which time the program returned to the immediately preceding slide.

Reference: Study is still in progress.

Program: Object Memory Span

Experimenter: James G. Holland

Programmer: Marjorie Jackson

Purpose: This program was essentially identical to the Digit Memory Span program described previously with the exception that pictures and names of everyday objects were substituted for numerals. The early frames in the program presented related objects and used stress and intonation to help the child retain the sequence. Later frames presented unrelated objects with equal stress and intonation.

Subjects: Pre-school age children

Program Characteristics: The functions of the Object Memory program were identical to those of the Digit Memory Span program with the exception that while only one slide was required for the Digit program, the Object Memory program required that a series of slides be coordinated with the appropriate audio messages. The final version of the Object Memory Span program was revised to include control of the Digit Memory Span experiment as well.

Reference: Study is still in progress.

Program: Logical Classification and Concepts of Relationship (LC&C)

Experimenter: James G. Holland

Programmer: Robert Fitzhugh

Purpose: This was another of the experimental CAI programs used in the project on training pre-school skills. It was designed to further the development of "formal logic," i. e., the abilities to form class concepts and to understand the basic relationships that exist between classes or between objects within a class. The material was designed to teach the child to match on the basis of (1) some identical perceptual quality, (2) some similar quality or qualities not readily perceived, and (3) use or function. It also attempted to teach the child to shift readily from one basis of classification to another, to induce classes from given groups of examples, and to deduce the reason for a given categorization.

Subjects: Children ages five to six.

Program Characteristics: The program was presented by use of the Touch-Sensitive Surface, one slide projector, and a random-access audio unit. The experimenter controlled the program by means of a teletype. Prior to an experimental run, the experimenter entered specific data about the subject (name, age, etc.), specified the starting item number, and typed in any additional comments he might wish to add about that particular subject. These comments were listed on the final data output for that subject.

The program was composed of five basic item types, the first four of which had similar program characteristics. These four were as follows: (1) A collection of objects was displayed on the screen all of which had one common attribute. The subject heard an audio message which named the attribute and asked him to select the one object

from a second array (shown on the bottom of the screen) which also had this attribute. (2) The subject was shown a collection of objects all having one common attribute. He was then shown another collection of objects and asked to touch as many objects as he could which had this same common attribute. (3) Shown an array of objects, the subject was asked to touch the one object which was different. (4) The subject was shown a collection of mixed objects and was asked to touch those objects in the collection which all had the same specified attribute. Touch surface areas on which no stimulus was projected were treated as null areas and touch responses in these areas were ignored. Any other response, correct or incorrect, generated a short high-pitched audio tone to indicate that the response was registered. A correct response was reinforced by a light and a loud tone. If the correct response was also the subject's first response to that item, he was further rewarded with a marble. A subject continued working on a frame until he made the correct response. If one or more incorrect responses were made on a frame, the program re-presented the previous frame.

The final, type-5, items are of particular interest. The subject was shown a collection of objects and asked to categorize the objects in as many different ways as he could. Any one slide contained several different sets of objects. That is, the objects might be categorized on the basis of size, color, function, etc. Any one object might be a member of several different sets. As the child responded, his successive responses defined the category with which he was working, e. g., large, small, blue, red, headgear, footwear, etc. If the subject deviated from a category which he had defined by his previous responses, failed to find all the objects in that category, or started to repeat a category, he was told the nature of his error and required to begin that category again. He continued working on a frame until he had exhausted all of the categories (two to eight) available on that slide.

The logic of this section of the program is of special interest since it demonstrates the programming efficiency that may be achieved by means of bit manipulation and the powerful decision-making capacity of a carefully designed program. Each type-5 frame had 10 to 15 data words associated with it, one for each object on the slide. Each possible attribute was represented by a bit-position in that word. If an object had a particular attribute, the appropriate bit was set to one in that object's data word. An additional bit indicated whether or not that object had been touched while the subject was working on the current category. While a subject was working on a particular frame, data concerning that frame was stored in three registers: Word A--the set of attributes that the subject has defined for the current category by his responses thus far; Word B--the attributes of all previously completed categories in this frame; and Word C--the attributes of all of the categories in this frame.

The logical steps in the type-5 item routine are given below:

1. **Play the audio message, "Touch all of the things that go together."**
2. **Set the contents of Word A to zero and zero all of the "touched" flags in the data words.**
3. **Activate the touch surface with a conditional time delay so as to allow a response within a limited time period.**
4. **Was a response made, as opposed to the termination of the time delay?**
   If yes, go to step 6.
   If no, continue.
5. **Has the subject made any response in attempting to define a new category, i.e., is Word A non-zero?**
   If yes, play a prompting audio message, "You didn't finish, start over." Go to step 2.
   If no, go to step 1.

6. Did the subject touch any object, as opposed to a null area?
   If yes, provide an audio "beep" to indicate that the response was registered. Continue.
   If no, go to step 3.
7. Has this object already been touched during the subject's definition of this category, i.e., has the data word been tagged as touched?
   If yes, go to step 3.
   If no, continue.
8. Tag this object as touched by setting a bit in its data word. Is this the subject's first response in defining a new category, i.e., is Word A zero?
   If yes, store the attributes of this object in Word A. Go to step 10.
   If no, continue.
9. Does this object possess any of the attributes in the set of attributes defined by all previous responses in this category? This is determined by anding this object's data word with Word A.
   If yes, store the result in Word A and continue.
   If no, play the audio message, "Those don't go together." Go to step 1.
10. Are any of the current set of attributes different than the attributes of the completed categories. And the contents of Word A with the complement of Word B.
    If yes, continue.
    If no, whatever attribute the subject is working on has already been done. Play the audio message, "You already did that one, start over." Go to step 2.
11. Has the number of attributes been reduced to one by the subject's series of responses? That is, is only one bit set in Word A?
    If yes, continue.
    If no, go to step 3.

12. Have all objects possessing this attribute been touched? Is the "touched" bit set in all of the data words in which the bit corresponding to this attribute is set?
    If yes, continue.
    If no, go to step 3.
13. This category is completed. Present the reinforcement stimuli, add the new attribute bit to Word B, and continue.
14. Have all categories on this slide been exhausted? And the complement of Word B with Word C.
    If yes, go on to the next frame.
    If no, go to step 1.

Data were stored on disc as the program ran and punched out at the end of the run by a background job. The resultant paper tape was later used as input to a FORTRAN program which reduced and listed the data.

Reference: Experiment still in progress.

Program: Simple Spelling

Experimenter: Merryl Samuels

Programmer: Raymond McKnight

Purpose: Simple Spelling was designed to investigate a specific question concerning the treatment of errors in a spelling program and the effect of two treatments on the subject's long and short term memory of misspelled words. Under the first condition, the subject was required to copy the correct spelling of an incorrectly spelled word. Under the second condition, the subject was simply told to study the correct spelling of an incorrectly spelled word. In both cases, the child was then re-tested on the spelling of the word. All words were then reviewed at the end of that day's session and following a period of one week.

Subjects: Children six to eight years of age

Program Characteristics: Instructions and the spelling words were presented via the random-access audio units. The subject responded by typing on a teletype which was also used by the teacher to control the program. Each word was first presented by an audio message such as "Spell the word Sunday." The phrase "Spell the word -" was recorded on the belt only once. After playing this message, the program selected and played the specific spelling word. As the child typed, his response was checked letter by letter. If he typed an incorrect letter, that letter was not printed and he was given a chance to correct his mistake by re-typing the letter. He was allowed a total of three errors in any one word before that word was counted as incorrect.

When a word was deemed incorrect, the child received either the copy or study treatment on a random basis. Under the copy treatment, the correct spelling of the word was typed out and the child was instructed to copy the word. Any copying errors caused the word to be re-typed and the copy message to be repeated. If the child made more than two copying errors, the program went on to the next word. If the word was copied correctly, the teleprinter paper was moved up to hide the correct spelling of the word and the child was re-instructed to try typing the word. The program then moved on to the next word regardless of the correctness of his response. Under the study treatment, the correct spelling of the word was displayed and the child was instructed to study the correct spelling. After four seconds, the paper was shifted up and the child was re-instructed to type the word. Again, the program moved on to the next spelling word regardless of his response.

Each day's session consisted of 15 spelling words. At the end of a daily session, the child was tested on all words that he had received that day and on the words which he had received a week earlier.

Reference: Pilot study.

On Developing a Computer Laboratory

This chapter will attempt to summarize some of the experience gained during the development and use of the LRDC Computer Facility. Some of the points mentioned may be more or less obvious, but taken as whole, they may be of value to the interested behavioral experimenter. It must be emphasized that these conclusions are based on a single sample and as such have limitations, particularly if one is interested in extrapolating the implications to larger or smaller systems. Also, it should be noted that the author has been more concerned with the software aspects of system development than with the system hardware, and this has undoubtedly biased his viewpoint.

The type of system described in this paper is only one approach to on-line behavioral experimentation. An individual experimenter may find a very small, non-time-shared machine, capable of running one subject at a time, to be a more satisfactory approach and, more importantly, within the limits of his resources. In some cases, terminals attached to university or commercial time-sharing systems may be adequate. While this is usually a much less expensive procedure, it must be recognized that the degree of experimental control is substantially reduced; this appears to be the case, at least, with the time-sharing services currently available. In the opinion of the author, the type of system described in this paper is the optimal solution for a research center or university department which includes a number of interested experimenters and has available resources which are sufficient for developing and maintaining such a system.

Given that it is feasible and desirable to develop a medium- to large-scale time-shared system, how does the interested but relatively naive researcher set about the task? Unfortunately, there do not appear to be any completely satisfactory answers. While the researcher should become as well-grounded in computer technology as is possible for him, it is doubtful that he can become as competent as he would wish without neglecting his research interests. Reading Uttal's Real Time Computers (1968) and Green's Digital Computers in Research (1963) is probably one of the best ways to gain an initial exposure to the area. There are several operating on-line laboratories in the U.S. and Canada, and it would be well to investigate the strengths and weaknesses of as many as possible (e.g., University of California, La Jolla; University of Colorado, Boulder; Ontario Institute for Studies in Education, Toronto; University of Texas, Austin; Stanford University; University of Rochester).

Since the researcher probably cannot become as knowledgeable as he would desire, it is obvious that he will have to seek expert advice and assistance. In a field as new as that of time-shared computer systems, however, it is not always easy to identify an expert, and the advice received from various quarters is likely to be quite different. One solution would be to rely on a single manufacturer to analyze requirements and to suggest a particular system configuration, but this has its obvious disadvantages. The researcher can only hope to become sufficiently knowledgeable to be able to make an intelligent choice between the alternatives suggested to him. If it is at all possible, it would be preferable to obtain the services of a competent and experienced computer technologist in the earliest stages of the project. Such help might be made available from the university computer science department, but there are distinct advantages to having a computer facility manager or consultant whose primary interests lie within the facility.

Installation and Development

The installation of a medium-sized computer system is likely the largest single capital investment that a behavioral science laboratory will make. While it is obvious that there is more than one computer manufacturer in this country, it is essential that the manufacturer(s) selected to supply the equipment be competent. It must be recognized that, at this time, time-sharing is still a state-of-the-art affair and even the best companies have not had unqualified success with their time-sharing systems. Beware of the manufacturer who offers a particularly attractive price tag because, as he states, "The company is trying to break into the field." A bargain system or bargain components may well turn out to be extremely expensive in the long run, due to their low reliability and, in some cases, actual design errors. Reliability, particularly in devices requiring mechanical movement, should be a prime consideration.

While the LRDC system is composed of units manufactured by several different companies and does not necessarily suffer from this heterogeneity, too wide a variety of manufacturers can raise problems of compatibility among devices. At this time, there is relatively little standardization in the computer field, and the mismatching of system components can cause a serious degradation of the system as a whole. When a manufacturer has been selected for a particular device, be sure that the standards for acceptance are clearly understood by both parties and that the acceptance tests themselves are clearly specified in advance. If at all possible, the actual installation of components should be overseen by a competent engineer associated with the facility.

It is quite likely that the cost of preparing the software necessary for the operation of the system will at least equal the investment in hardware. Software preparation is inevitably slower than anticipated. In addition to the visible costs of personnel salaries, one must consider the costs of the delay before the system can be placed on a production basis. Consequently, it is advisable that as much of the manufacturer-supplied software be utilized as is possible. When LRDC began the development of its current system, there was very little software available which was suitable for the laboratory's requirements. Consequently, the time-sharing system was developed from scratch. More recently, the Facility has been able to make extensive use of the newer software supplied by the manufacturer. In most cases, it has been necessary to modify the manufacturer's software to make it more compatible with the requirements of the system, but the cost of modification is minimal as compared with the expense of the complete development of similar software. When software is developed in-house, compatibility with the manufacturer's software should always be kept in mind.

Be pessimistic about the capability of your systems programming staff. Despite its difficulty, or perhaps because of it, systems software design and development is extremely interesting and challenging work. Consequently, systems programmers often tend to promote approaches which, while they are indeed feasible, are beyond the capacity of the available programming staff. While a slightly more sophisticated and elegant routine may be more appealing, it may not be at all justified in terms of the extra effort required for its implementation.

If extensive system software development is required, a substantial period of time will elapse before the system is fully operational. Although much of the system design and programming can and should be done before the hardware is installed, the bulk of the work requires that the equipment be installed and operating. There are at least three approaches to implementing a time-sharing system. First, all use of the system can be delayed until the final system is fully operational. While

this approach eliminates the majority of the headaches, it is not feasible except under the most unusual circumstances. Assuming that the desired system is fairly sophisticated, a significant period of time will be required for its implementation. Very few laboratories can justify the time and expense expended before the system can be used. Furthermore, it is extremely unlikely that a completely satisfactory system can be designed without feedback from the researchers who will be using it.

A second alternative, and the approach taken by the LRDC Facility, is to implement the skeleton of the desired system as soon as possible. This allows the system to be used in at least a limited manner within a much shorter period. The system is then expanded and refined as it is being used. Although the production capacity of the skeleton system falls far short of what is desirable, some form of a system is, at least, available to those experimenters who are willing to tolerate its shortcomings. The experience acquired in using the system for experimental control is quite valuable for its further development. This course of action does have definite disadvantages, however; since the system is constantly changing, control programs for particular experiments quickly become obsolete. A considerable amount of effort is expended in implementing the abbreviated routines which then have to be extensively revised at a later date. Since the system is constantly being modified, reliability cannot be maintained at a desirable level. In general, this approach seems to lead to a situation in which the efforts of the staff are constantly being diverted to fight brush fires--making relatively minor but urgently needed and time-consuming modifications to the system. As a consequence, the overall development of the full system is seriously retarded and production is held at a low level unduly long.

With the advantages of hindsight, a third, and more satisfactory course of action, would appear to be to begin by implementing a very limited, probably non-time-shared system that fulfills as many of the laboratory's experimental requirements as possible. The effort required to implement even such a minimal system is negligible, but it appears to be the shortest route to placing the laboratory on some sort of an operational basis. Once the minimal system is operating at a satisfactory level, an advanced time-shared system could be developed. A limited but stable system would be available for research. A programming staff could be developed and trained during the implementation of the first system, and they would have gained experience in working with this particular equipment. Use of the limited system may point up problems which would be much more serious if they were discovered in the more advanced system. Most importantly, work on the advanced system could proceed much more rapidly if the maintenance of the currently operating system does not require the continual attention of the staff.

Despite the two-step implementation plan described, it would be quite unusual if the advanced system discussed became the "final" system. System modifications and refinements always seem to be necessary. Subject terminals and I/O devices will be replaced as new components become available. It is very likely that additional core or mass storage capacity will be required and that the system in general will be upgraded as utilization increases. Such alterations to the system will cause disruptions of the research program but these disruptions can be minimized. When new components are added to the system, the required down-time can be kept to a minimum if the interface between the new component and the system is completely designed prior to installation. Delivery time on most computer support equipment is sufficiently long that the new software required can be prepared and ready for testing as soon as the device is installed. When new devices are added, every effort should be made to make them appear to the system to be as similar as possible to existing devices. There are obvious limits to this strategy as when, for example, it limits the efficiency of the total system, but system stability should not be sacrificed for a minor point of hardware design elegance. Often, the software device control routines can adapt to essential hardware modifications without altering the command structure used by the experimental control programs in calling the routine.

One of the major advantages of the LRDC system, and of any sufficiently flexible on-line system, is the variety of terminal devices which it can support. In the long run, it is less expensive and more productive to purchase off-the-shelf items if suitable devices are available but the in-house construction of terminal devices tailored to an experimenter's specific requirements can often be quite rewarding. Most behavioral science laboratories do not have the resources to conduct an extensive engineering research and development program, and the number and scope of such projects should be limited accordingly. Again, new devices should be designed so that their interface with the system is as similar as possible to that of existing, standard devices. Be pessimistic about the reliability of a new device when it is first installed into the system. The researcher who requested the device will undoubtedly want to use it but his experiment should be treated as a special, non-production class of run for which it is recognized that the usual level of system reliability cannot be guaranteed. Any such experiment should not be tied into a tight time schedule. If it is, the emergencies which arise will have an unduly detrimental effect on the research of the other experimenters using the system.

When software modifications are required, strive for upward compatibility. That is, programs which ran under the previous system should be able to run under the modified system without revision. There is a strong temptation to implement any system improvements as soon as they are available. In general, however, it is more efficient to update the system in phases rather than piecemeal, i. e., a number of modifications can be collected and implemented at one time in what is recognized as being a substantially revised system. This procedure is particularly important if one or more of the modifications did not preserve upward compatibility. While any system modification must be tested extensively prior to implementation, there will always be a greater chance of errors immediately following a system modification. A limited number of major system revisions will mean that system reliability will be substantially decreased for short periods of time but this is more desirable than having the system reliability continually degraded by frequent, minor revisions. When the system reaches a fairly stable point, that is, when the systems staff begins to catch up with its work load, it would be well to concentrate on refining the system. Unless they have been completely planned in advance, systems tend to grow like Topsy. Take a hard look at the system characteristics and options which are seldom used. Are they really necessary? Is there redundancy in the system which could be eliminated?

Personnel and Management

The most important component of a productive on-line facility is its staff. It will be no surprise to anyone remotely concerned with the computer field to be told that attracting and keeping a competent staff is a definite problem. The field has grown so quickly that there is a serious shortage of personnel in all areas of computer applications but particularly in programming. This is an especially serious problem for installations in a university setting which do not usually have the financial resources to compete with business and industry. The university installation does have the advantage, however, of being able to attract employees who are interested in furthering their education. Over half of the current LRDC Computer Facility personnel are working toward undergraduate or advanced degrees. About a quarter of these are full-time students employed on a part-time basis but the majority are full-time employees who are given some latitude in their working hours so as to attend a limited number of classes.

Many university-based installations make extensive use of graduate students as programmers, technicians, or engineers. This has the advantage of providing an intelligent and often highly skilled staff at a relatively low cost. However, it must be recognized that graduate and undergraduate students are students first and employees second. As a rule, they are not able to work regular hours and, consequently, they are unsuitable for supervisory positions. Inevitably, they are unavailable if an emergency develops in their area of specialization. Due to the demands of their studies, their work often comes in bursts separated by periods of low or negligible production. This makes it very difficult to integrate their work into a larger staff effort.

It is possible to capitalize on the talent of student personnel, however. The central core of the LRDC Facility staff and all of the supervisors are full-time employees. Students employed on a part-time basis are assigned specific problems in their area of specialization which are not crucial to the immediate operations of the Facility. In one case, a graduate student, who is very familiar with the Central Executive software, is used, essentially, as a systems programming consultant. No students are used as Applications Programmers since it is in this area that the pressures of scheduling and prompt delivery are the most severe. Under this policy, the Facility benefits from the particular skills of these students but is not hampered by the difficulty of placing them on rigid time schedules.

One seemingly appealing means of extending the capacity of a limited staff is by employing consultants or by contracting work to a software house. While this may be profitable under some circumstances, it is far from being a panacea. Consultants are expensive and the good ones are usually very expensive. It is suggested that the best way to use consultants is to have them in for short, one-shot sessions for the purpose of evaluating the system or generating ideas which can be carried out by the facility staff. This procedure assumes, of course, that the consultant has been well supplied with systems documentation and information about the problem prior to his visit. If software must be contracted out, it is preferable to limit the contracts to specific programming tasks which do not form a central part of the time-sharing system and are not critical for immediate operations. Once the software is installed, the facility staff will have to maintain and modify it. Working with someone else's program is never easy and the task will be still more difficult if the software does not adhere to the programming and documentation standards current in the facility.

There is the problem of the optimal allocation of the available funds. In the opinion of the author, at least, the managerial positions should be the most competitive with industry. If the management and supervision is sufficiently skilled, productive use can be made of more trainee-level employees. It is a particular bias of the author that the manager of a computer facility supporting a research laboratory should be a computer specialist, preferably with an industrial background, and not a researcher. While hopefully, the researcher's work will benefit from the use of the facility, its development and management is not a means by which he can advance in his field. Consequently, the problems of the facility must compete with his other commitments in teaching and research. Furthermore, the researcher will not have the experience and skills which a computer specialist can bring to the job. Managing a facility the size of the LRDC installation is a full-time job requiring all of the talents that the specialist can bring to bear. As is discussed below, the procedures which seem to be necessary to develop a productive computer facility are at variance with the approach taken to research in an academic setting. An industrially trained computer specialist is more likely to institute the type of strict organization which appears to be required for the facility to provide adequate service to the experimenters who are its customers. This is not to imply that the manager be given the authority to determine the functions of the facility--this is the responsibility of the research staff who are the users of the facility--but he should be given full authority for determining the operating policies of the facility and given a full voice in determining the direction of future development. The manager can also obtain additional recognition by carrying out research and development in his own field, if he so desires.

The justification for a computer system in a research laboratory is the service which it provides the experimenters. The system is not a toy or a testbed for the ideas of the engineers and systems programmers. Whenever a modification to the system is proposed, it must be evaluated on the basis of the degree to which it will improve research production. The only acceptable basis for judging the system's value is the quantity and quality of the research which it supports. Thus, the utility of the system is the joint responsibility of the facility management and their users, the experimenters.

The development of a time-shared, on-line system is a state-of-the-art endeavor. The problems are real but not insurmountable. The effective operation of a facility, like the LRDC installation, requires efficient cooperation among a number of specialists. Although the idea is distasteful to most academically oriented researchers, it is the author's opinion that to be most effective, the facility staff must be run as a heavily organized and tightly scheduled team. It is often accepted as a matter of fact that software development takes longer than the time allotted for it. The same often holds for hardware modification and repairs. It is only by means of a tightly organized system of scheduling and frequent progress reports that the integrated development of a system can proceed on anything like a regular basis.

As was mentioned above, much of the responsibility for the success of an on-line laboratory lies with the experimenters who use it. Simply because the system is available does not mean that it should be used for all experiments. In many cases the system provides no advantage over flash cards or a memory drum, and its use serves only to retard the work on more suitable research. Experimenters should be encouraged to specify their long-range research plans in as much detail as possible. This will enable the applications programming staff to develop or at least design a package of programs which can control a series of experiments with only minor modifications. This is much less time-consuming and expensive than a series of one-shot experimental programs. Whenever possible, the experimenters as a group, and the facility staff should agree on standardized formats for the input and output of experimental parameters and data production.

In the LRDC laboratory, it is unusual for an experimenter to write his own control programs. This is partially due to the complex nature of the system and the current lack of a satisfactory higher level language, but even if this were not the case, the author would recommend that the experimenter turn over his programming tasks to an applications programming group. The behavioral researcher is a specialist in his own field and that field is not computer programming. In some disciplines, such as mathematics, there is such a serious communications barrier between the scientist and the programmer that it is often easier for the scientist to do his own programming than to attempt to communicate his ideas to a programmer. This is not the case in behavioral experimentation. After investigating several approaches at LRDC, the most satisfactory appears to be that of having the researchers communicate their programming requirements to one articulate skilled Programmer, in this case, the Supervisor of Applications Programming. The Supervisor does relatively little programming himself but has become a specialist in communicating with the experimenters and in the design of experimental control programs. He obtains his information by means of rudimentory flowcharts and discussions with the experimenter and/or the experimenter's staff. In the course of these discussions he is in a position to make suggestions about the optimal use of the different terminal devices, being familiar with the strengths and weaknesses of each, and at times suggests alterations in the experimental procedure to capitalize on the benefits of the system capabilities. On the basis of these discussions and materials, he designs the program by producing the detailed flowcharts from which the members of his staff work. Once the programming is underway, experimenters inevitably discover aspects of the experimental procedure which they wish to change. Since the Supervisor is familiar with the nature of the program, its current status, and the ability of the particular Programmer, he is able to give the experimenter a fairly accurate estimate of the additional time required to make the requested alterations.

I fear that I may have painted an overly-cautious, if not bleak, picture of the development of an on-line laboratory. This is not at all my

intention. The reader should be aware that such a project is a major undertaking and not one to be entered into lightly. There are dangerous pitfalls, particularly in the area of management, which can seriously curtail the productivity of the laboratory. While it is almost certain that some mistakes will be made, they can be held to a minimum. Correctly planned, developed, and managed, a computer-based laboratory can be a very rewarding venture. All in all, the potential benefits of on-line experimentation are well worth the effort.

REFERENCES

Bright, G. W. Scheduling and memory allocation algorithms for University of Pittsburgh computer based instruction system. Research Report 65-5KO-TEACH-R1. Pittsburgh, Pa.: Westinghouse Electric Corp., 1965.

Broadley, W. H. Utilities. 4. CORFIL, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Buckwalter, J. T. Systems. 3. CARD, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1967.

Buckwalter, J. T. Systems. 3. GRAB, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1966.

Buckwalter, J. T. Systems. 3. MAGTAP, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1967.

Buckwalter, J. T. Systems. 3. PUNZIT, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Buckwalter, J. T. Systems. 3. RENTRY, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1966.

Buckwalter, J. T. Systems. 3. SCREEN, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Buckwalter, J. T. Systems. 3. SYSERR, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1966.

Chadwick, M., & Fitzhugh, R. J. A user's guide to the LRDC integrated MACRO package. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, in preparation.

Cook, C. Systems. 3. KBUNLK, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1966.

Digital Equipment Corporation. FORTRAN IV, Programmer's reference manual, PDP-9 advanced software system. Maynard, Mass.: Author, 1968a.

Digital Equipment Corporation. I/O monitor guide for paper tape systems, PDP-9 advanced software system. Maynard, Mass.: Author, 1968b.

Digital Equipment Corporation. MACRO-9 assembler, programmer's reference manual, PDP-9 advanced software system. Maynard, Mass.: Author, 1967.

Fitzhugh, R. J. Utilities. 4. DEBUG, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Fitzhugh, R. J., & Katsuki, D. The touch-sensitive screen as a flexible response device in CAI and behavioral research. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, in preparation.

Glaser, R., Ramage, W., & Lipson, J. The interface between student and subject matter. Technical Report 5. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1964.

Glaser, R., & Ramage, W. W. The student-machine interface in instruction. 1967 IEEE International Convention Record, Part 10, New York: Institute for Electrical and Electronics Engineers, 1967, pp. 52-59.

Green, B. F. Digital computers in research. New York: McGraw-Hill, 1963.

Jackson, M. Systems. 3. DPHONE, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1968.

Jackson, M. Systems. 3. CROW, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Jackson, M. Systems. 3. DOODLE, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Jackson, M. Systems. 3. SOCK, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1968.

Judd, W. A. Systems. 3. MEMAL, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1967.

Judd, W. A. Systems. 3. WAIT, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1967.

Judd, W. A., & Glaser, R. Variability of response latency in paired-associate learning as a function of training procedure. Technical Report 9. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1970.

Judd, W. A., & Glaser, R. Response latency as a function of training methods, information level, acquisition and overlearning. Journal of Educational Psychology Monograph Supplement, **1969,** 60, **No. 4, part 2.**

Kaupe, A. F. Operating software for a computer-based instruction system. **Technical Report 3. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1966.**

Nemitz, B. P. SKOOLBOL: A simplified user's language for programming the PDP-7. Working manual. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1968.

O'Malley, J. M., Glaser, R., & Resnick, L. Response discrimination pretraining and list length in learning a multiple discrimination. Report in preparation.

Pethia, R. D. Systems. 3. DPHONE, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Pethia, R. D. Systems. 3. HYSPRJ, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Pethia, R. D. Systems. 3. TOUCHE, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1968.

Pethia, R. D. Utilities. 4. SYSLOD, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Ramage, W. W. Language properties for computer-assisted instruction. Conference notes. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1967.

Ramage, W. W. A mathematical investigation of three attributes of automated instruction. Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, in preparation.

Skillen, R. J. Utilities. 4. RECORD, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Slaughter, B. G. Systems. 3. DISC, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1968.

Slaughter, B. G. Systems. 3. PRINTR, Computer facility documentation, Pittsburgh, Pa.: Learning Research and Development Center, University of Pittsburgh, 1969.

Tobias, S. & Glaser, R. Effect of pupil choice of instructional method on achievement and attitude. Manuscript in preparation.

Uttal, W. R. Real-time computers: Techniques and applications in the psychological sciences. New York: Harper & Row, 1968.

3

APPENDIX A

Index of Computer Facility Documentation

MARCH 2, 1970

.INDEX.0.

.INTRODUCTION AND GENERAL DESCRIPTION.1.
    .1.APPLICATIONS DOCUMENTATION
    .1.DESCRIPTION OF THE CURRENT LRDC COMPUTER FACILITY SYSTEM
    .1.FLOWCHARTS
    .1.LRDC COMPUTER-BASED INSTRUCTIONAL SYSTEM CENTRAL EXECUTIVE
    .1.SYSTEMS DOCUMENTATION
    .1.SYSTEM EQUIVALENCES

.HARDWARE.2.
    **.2.CARD READER
    *.2.DATAPHONE
    .2.DISC
    .2.INBUS
    .2.INFORMATION COLLECTOR
    .2.7/9 INTERFACE
    .2.INTERRUPT
    .2.IOT
    .2.KEYBOARD
    .2.MAG TAPE
    .2.PRINTER
    .2.RA PROJECTORS
    .2.RELAY BUFFER
    .2.TELETYPE
    .2.TOUCH
    .2.TRAP

*DOCUMENTATION NOT YET AVAILABLE
**DOCUMENTATION NOT ACCURATE

.SYSTEMS.3.

  .EXECUTIVE PACKAGES
    .3.EXEC III

  .CENTRAL EXECUTIVE.
    .3.CALGON
    .3.COMMON
        SETCOM
    .3.CONVRT
    *.3.COUNT
    *.3.CREATE
    .3.CITY
    .3.DATIME
    .3.DEDTIM
    **.3.DZMROU

- *.3.FIND
- *.3.GETJB
- .3.GREASCAN
    - GRAB
    - RELEAS
    - SCAN
- *.3.INSERT
- .3.IOINT
- *.3.KILL
- .3.LOCATE
- **.3.MASTER TABLES
- .3.MEMAL
- *.3.NEWLST
- .3.OCTSCN
- *.3.PUTES
- .3.RELAY BUFFERS - APPLICATIONS
    - CLRIMG
    - CNGIMG
    - SETIMG
- *.3.RELAY BUFFERS - SYSTEMS
    - CLRIMG
    - CNGIMG
    - SETIMG
- *.3.REMOVE
- .3.RENTRY
- **.3.SETUP
- **.3.SUSPEN
- .3.SYMSCN
- .3.SYSERR
- **.3.TAKE
- .3.TARRY
- **.3.TIMDAY
- .3.TIMSTR
- .3.WAIT

*DOCUMENTATION NOT YET AVAILABLE
**DOCUMENTATION NOT ACCURATE

.PERIPHERAL EQUIPMENT ROUTINES
- .3.BAGTEL
- .3.CARD
- .3.CROW
- .3.DISKUS
- .3.DPHONE
- .3.HYSPRJ
- .3.INVENT
- .3.KBUNLK
- .3.LASTFL
- .3.MONTAP
- *.3.MTAPE
- .3.NEWTAP
- .3.PRINTR

- .3.PRJSET
- .3.PUNCH
- .3.PUNZIT
- .3.SCREEN
- .3.SEEK
- .3.TOUCHE

*DOCUMENTATION NOT YET AVAILABLE

.OPERATOR CONTROL ROUTINES.

- .3.DOODLE
  - .SOCK.DRED
  - .SOCK.DWRT
  - .SOCK.DDEL
  - .SOCK.DLST
- .3.SOCK
- .3.SOCK.ANFG
- .3.SOCK.CNAD.CNDL
- .3.SOCK.CRET
- .3.SOCK.JBGO
- .3.SOCK.JOBZ
- .3.SOCK.KILL
- .3.SOCK.RELS
- .3.SOCK.SETP
- .3.SOCK.SUSP
- .3.SYSDMP
  - .SOCK.DUMP

.UTILITIES.4.

- .4.ASMLOD
- .4.ASMWRT
- .4.ASPRIN
- .4.BCVTI2
- .4.BINTAPE DUMP
- .4.CORFIL
- .4.DISCRD
- .4.DISDMP
- .4.DISK EXERCISER
- .4.DISK TEST ROUTINE
- .4.DISK9
- .4.DUMP
- .4.DUMPTY
- .4.FILPLA
- .4.FLSTAT
- .4.FOMI02
- .4.FOMI04
- .4.LODUMP
- .4.MLOP
- .4.OCLMT
- .4.OCTAL CARD LOADER
- .4.PMD
- .4.PRINTER

.4.RECORD
.4.SWAP
.4.SYSLOD
.4.SYSWRT
.4.TAPE SPLASH
.4.TND3
.4.90T09
*DOCUMENTATION NOT YET AVAILABLE

.DIAGNOSTICS.5.
.5.CLOCK
.5.CRT
.5.DISCHK
.5.MAGTAPE
.5.PMDTST
*.5.PRINTER ACCEPTANCE TEST
.5.PRINTER ALIGNMENT
.5.TRAPT
*DOCUMENTATION NOT YET AVAILABLE

.SERVICE ROUTINES.6.
*.6.LINEDRAW
*DOCUMENTATION NOT YET AVAILABLE

.MANUFACTURER SUPPLIED SOFTWARE.7.
.7.PDP-7 USERS HANDBOOK
.7.PDP-9 USERS HANDBOOK (DEC-F-95)
.7.PDP-9 MACRO ASSEMBLER (DEC-9A-AM9B-D)
.7.PDP-9 FORTRAN IV (DEC-9A-KFZA-D)
.7.PDP-9 MONITORS (DEC-9A-MAB0-D)
.7.PDP-9 I/O MONITOR GUIDE FOR PAPER TAPE SYSTEMS (DEC-9A-NGAA-D)
.7.PDP-9 UTILITY PROGRAMS (DEC-9A-GUAB-D)
.7.PUNCH9

.HIGH LEVEL LANGUAGES.8.
.8.SKOOLBOL - A SIMPLIFIED USERS LANGUAGE FOR PROGRAMMING THE PDP-7

.APPLICATIONS.9.
.9.ANGLE 6
.9.BIGSPELL/TTY
.9.FEEDBACK SPELLING NAME FILE UTILITY
.9.FEEDBACK/PHASE I DATA BACKUP UTILITIES
.9.LCCI
.9.LETTER DISCRIMINATION
.9.LETTER DISC. DATA
.9.LETTER DISC.DATA REDUCTION

.9.LETTER DISC. OUTPUT
.9.NUMBERS SECT 1
.9.NUMBERS SECT 2
*.9.NUMBERS SECT 3
.9.NUMBERS 1 OUTPUT PROGRAM
*.9.NUMBERS 2 OUTPUT PROGRAM
.9.PHASE 1 SPELLING NAME FILE UTILITY
.9.PREFERENCES
.9.PREFERENCES DATA REDUCTION
.9.PREFERENCES OUTPUT
.9.TAP II, CONDITION I
.9.TAP II, CONDITION II
.9.TAP II, CONDITION III
.9.TAPOUT

*DOCUMENTATION NOT YET AVAILABLE

APPENDIX B

LRDC COMPUTER FACILITY

APPLICATIONS PROGRAMMING GROUP

DOCUMENTATION STANDARDS MANUAL

CONTENTS

I. INTRODUCTION

II. APPLICATIONS DOCUMENTATION OUTLINE AND EXPLANATION

III. EXAMPLE

  A. LCCI

I. INTRODUCTION

The following outline is designed to assist applications programmers when documenting a program. In writing any documentation, the programmer should attempt to completely cover all aspects of the program in a clear and concise manner. The documentation should contain enough information to:

a. uniquely identify the program
b. explain the functions it performs, and
c. enable others to use the program.

It is also necessary that applications documentation contain enough information so that other programmers may understand the logic and flow of the program and to expand or modify the program, if necessary.

In summary, the chief objective of any documentation should be to clearly describe and explain all aspects of the program's structure, use, and operation. The documentation for LCC (.9.LCCI, 10/10/69) clearly illustrates the documentation procedures described on the following pages.

II. APPLICATIONS DOCUMENTATION OUTLINE AND EXPLANATION

1. PROGRAM NAME:

   Contents: Name of project.

2. PROGRAM I.D.:

   Contents: Identification code for program.

3. REQUESTED BY:

   Contents: Name of user who requested program.

4. USER REPRESENTATIVE:

   Contents: Name of person to whom questions should be directed.

5. DATE:

   Contents: Date of documentation.

6. PROGRAMMER(S):

   Contents: Names of original programmer(s).

7. MACHINE:

   Contents: The basic machine on which the program was designed to run.

8. LANGUAGE:

   Contents: The programming language in which the program was written.

9. ABSTRACT:

   Contents: A brief description of the main function of the program -- what it does and how it fits into the system.

10. HARDWARE REQUIREMENTS:

    Contents: A list of the I/O devices used by the program.

11. SOFTWARE REQUIREMENTS:

    11.1 EXEC VERSION:

    Contents: Identification of Executive system for which the program was written.

11.2 PERP ROUTINES REQUIRED:

Contents: Listing of the PERP routines the program requires.

11.3 MEMAL REQUIREMENTS:

Contents: Listing of the memal blocks used in the program.

11.4 GRABS AND RELEASES:

Contents: Indication of when I/O devices are grabbed and released. The devices should be specifically named.

11.5 PROGRAM SUSPENSIONS:

Contents: Every suspension in the program should be indicated with the amount of time or suspension reason identified for each.

11.6 MEMORY LAYOUT:

Contents: The attached appendix sheet should be filled out completely for all four fields starting with field zero.

All logical blocks should be identified. For example, the following should be shown: data, Common, Memal, mainline logic, and the literal or constant pool.

It will be assumed that all core locations not listed will be available for use by other programs.

11.7 AUXILIARY STORAGE REQUIREMENTS:

Contents: A listing of what storage other than core storage is required by the program. The listing should be by device type.

11.8 REENTRANT STATUS:

Contents: An indication of whether or not the program is time sharable and/or code sharable.

12. DESCRIPTION OF PROGRAM FLOW:

12.1 GENERAL INTRODUCTION AND PROGRAM PHILOSOPHY:

Contents: An introduction and explanation of the logical approach taken by the program. For example, if the program were data driven, that concept and its application to the program would be explained here.

12.2 NARRATIVE:

Contents: A verbal description of program flow in two parts. First, a description of the mainline flow. Secondly, a brief description of the paths through the subroutines. Any unique or special circumstances as well as input data structures should be explained in this section.

Reference should be made to symbolic locations in the program. This description should be written with the understanding that the narrative along with a listing of the program and a complete set of program flowcharts would provide enough information for a programmer not familiar with the program to follow the program flow and understand the main functions of the program.

12.3 FLOWCHARTS:

Contents: All significant subroutines and the mainline of the program should be flowcharted.

12.4 I/O RECORD DESCRIPTION:

Contents: The attached appendix sheet should be completed as follows:

DATA NAME: The name of the field as used in the program.

FORMAT: An indication of the field format, i.e., packed ASCII, sixbit, etc.

SIZE: The number of core locations for the field.

FREQUENCY OF USAGE: An indication of how frequently the contents of the field are used, i.e., once per frame, twice per frame, etc.

USAGE: How the contents of the field are used.

12.5 FILE DESCRIPTION:

Contents: The attached appendix sheet should be completed as follows:

FILE ID: File name used in the program.

DEVICE: Device on which the file resides.

RECORD LENGTH: Length of a single record in words.

BLOCKING FACTOR: Block size.

TOTAL # RECORDS: Exact and/or good approximation of the number of records in the file.

I/O: Indicate if file is input or output.

USAGE: Indicate how the information in the file is used.

13. COMPUTER OPERATOR PROCEDURES:

Contents: This section should describe all the operator procedures necessary to run the program(s) being documented. Any of the following subsections that are not applicable should simply be excluded. The name of each program should be included in parenthesis after each major category name.

13.1 INPUT DATA:

Contents: If there is any input data handling to be done by the operator, the following subsections should be completed. Again, if a subsection does not apply, exclude it.

13.11 DATA PREPARATION:

Contents: A description of, or instructions for, any physical preparation of input data by the operator. This may include mounting a specific magtape, placing cards in the card reader, placing paper tape in the reader or any other possible preparation.

13.12 HARDWARE SYSTEM PREPARATION:

Contents: A complete list of things to do with any of the hardware or peripheral devices to facilitate input data preparation.

13.13 PROGRAM LOADING PROCEDURES:

Contents: If a special program is used for data preparation, its loading instructions should be under this subtitle. There should also be instructions to load a backup program in case there is a failure in loading the primary program.

13.131 STANDARD PROCEDURE:

Contents: If the program is to be loaded from disk using DOODLE, the file name must be given along with any information concerning possible field dependence of the file.

13.132 BACKUP PROCEDURE:

Contents: When the program is to be read from magtape using the T & D TRIO, the tape number(s) and file number(s) must be given. Finally, if a program is on punched paper tape, the location of the tape tray, identification of the program tape(s), loading sequence of the tapes and any information concerning possible field dependence of the tapes.

13.14 SYSTEM INITIALIZATION:

Contents: A complete list of control teletype command examples necessary to set up and start the program. In constructing the examples, bear in mind that the job number and physical unit number are variable.

13.15 OPERATOR RUNNING PROCEDURES:

13.151 NORMAL OPERATION:

Contents: A clear description of, or instructions for, any thing the operator must do while running the input data preparation program.

13.152 PROGRAM/SYSTEM ERRORS:

Contents: Include possible sources of program failure and steps to recover from a possible failure.

13.16 SYSTEM TERMINATION:

Contents: A complete list of control teletype command examples necessary to terminate the input data program. Also some indication of how the operator would know when the program has normally terminated itself.

13.17 OUTPUT DATA HANDLING:

Contents: Clear instructions for whatever the operator should do with the output (if any) from the input data program.

13.2 MAIN PROGRAM:

Contents: A complete, precise description of the loading, operating and unloading of the main program.

13.21 HARDWARE SYSTEM PREPARATION:

13.211 CENTRAL PROCESSOR:

Contents: An indication of the necessary action other than the initialization of the time-sharing system.

13.212 STUDENT STATION:

Contents: A complete list of the things an operator must do to make a student station operational. Include such things as CROW belt numbers, slide trays, projectors, etc.

13.22 PROGRAM LOADING:

Contents: Clear instructions for the standard procedure and backup procedure for loading the main program. See 13.13 PROGRAM LOADING PROCEDURES.

13.23 SYSTEM INITIALIZATION:

Contents: A complete list of control teletype examples to set up and start execution of the main program.

13.24 OPERATOR RUNNING PROCEDURES:

Contents: See 13.15 OPERATOR RUNNING PROCEDURES for details.

13.25 SYSTEM TERMINATION:

Contents: See 13.16 SYSTEM TERMINATION for details.

13.3 OUTPUT DATA PROGRAM:

Contents: If no special program is necessary to process output data, then this section should only tell what is to be done with the output. Things such as labeling, routing, etc. should be here.

If a separately documented output program is used, this section should refer to that documentation. If the output program is included in this documentation, the following subsections should be used to clearly describe the program operating procedures. Reference is to previously defined items.

13.31 DATA PREPARATION:

Contents: See 13.11.

13.32 HARDWARE SYSTEM PREPARATION:

Contents: See 13.12.

13.33 PROGRAM LOADING:

Contents: See 13.13.

13.34 SYSTEM INITIALIZATION:

Contents: See 13.14.

13.35 OPERATOR RUNNING PROCEDURES:

Contents: See 13.15.

13.36 SYSTEM TERMINATION:

Contents: See 13.16.

13.37 OUTPUT DATA HANDLING:

Contents: An exact description of paper tape labels, output bins, etc., as they apply to this output data.

14. OPERATIONS PROCEDURES SUMMARY:

Contents: A summarization of operator procedures for a single run of the program.

14.1 INPUT DATA PROGRAM:

Contents: Input data program requirements including loading and termination.

14.2 TERMINAL:

Contents: A list of student station set up requirements.

14.3 PROGRAM LOADING:

Contents: Normal DOODLE request format.

14.4 INITIALIZATION AND START-UP:

Contents: Steps to begin execution of program.

14.5 RUNNING PROCEDURES:

Contents: Indicate any operator action required during run.

14.6 TERMINATION AND OUTPUT PROGRAM:

Contents: Steps to terminate main program and collect output data.

15. USER PROGRAM OPERATION PROCEDURES:

Contents: This section should clearly describe all the procedures necessary to operate the program as the user sees it. These instructions should be so written that even an untrained or unexpected user would have a minimum of difficulty operating the program.

15.1 INPUT DATA PREPARATION:

Contents: A description of input data preparation including items such as: Data Vocabulary and Data Syntax. Extreme detail and illustration with examples should be given.

15.2 START-UP:

Contents: Any steps necessary for the user to start or initialize the program. Include a complete list of messages given by the program for this purpose.

15.3 PROGRAM OPERATION:

Contents: A clear description of whatever the user must do during program operation. Again, include all possible teletype or CRT messages and an explanation of each of them.

15.4 TERMINATION:

Contents: An explanation of what the user must do to properly terminate the program. Include all messages to this effect.

15.5 PROGRAM LIMITATIONS:

Contents: An explanation of just what may or may not be done with the program while it is running. Include such things as warnings against improper start-up, termination, or restarting.

15.6 OUTPUT DATA HANDLING:

Contents: If there is a possibility of the user not having output processed immediately, this section should include an explanation of how the user may have the raw data processed at a later time. This section may also include an explanation of printed output, tape markings, or any other information concerning the output data.

15.7 POSSIBLE SOURCES OF PROGRAM FAILURE:

Contents: A list of all the things (within reason) the user might do that would cause a program failure. Include a description of how the user might know that the program is down.

15.8 RECOVERY FROM PROGRAM FAILURE:

Contents: Anything the user can do to recover from a program failure (if possible).

16. ADDITIONAL REFERENCES:

Contents: A listing of all reference material associated with the program should be included here.

-EXAMPLE-

11.6 MEMORY LAYOUT

PROGRAM: ______________________________ **ID:** ______________

FIELD	LOWER BD.	UPPER BOUND	CONTENT

-EXAMPLE-

12.4 I/O RECORD DESCRIPTION

PROGRAM: **ID:**

RECORD NAME: **LENGTH:** **I/O STATUS:**

INPUT TO: **OUTPUT TO:**

DATA NAME	FORMAT	SIZE	FREQUENCY OF USAGE	USAGE

-EXAMPLE-

12.5 FILE DESCRIPTION

PROGRAM: ______________________________ **ID:** ______________

FILE ID	DEVICE	RECORD LENGTH	BLOCKING FACTOR	TOTAL # RECORDS	I/O	USAGE